Pembuatan Prototipe Firewall pada Embedded PC Dengan Menggunakan FreeBSD

TUGAS AKHIR

Oleh :

Andias T. Wira A
5198.100.088

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
SURABAYA
2004

Pembuatan Prototipe Firewall pada Embedded PC dengan Menggunakan FreeBSD

TUGAS AKHIR

Diajukan untuk Memenuhi Persyaratan
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
SURABAYA

Menyetujui,

Dosen Pembimbing

Febriliyan Samopa, S.Kom, M.Kom
NIP. 132 206 858

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
SURABAYA
2004

Ich bin jetzt nicht erfolgreich.

Wenn ich mich ernsthaft bemühen würde,

hätte ich keine Schulden.

Nach meinem indonesichen Studium,

möchte ich eine Forschungreise machen.

Schließlich, will ich

«meinen Eltern"

«meiner Schwester"

«meiner Freundin"

dieses Buch widmen

ABSTRAK

Produk *firewall* komersial dalam bentuk *hardware* sudah banyak beredar di pasaran dan harganya pun mahal. Banyak orang di Indonesia bisa membuat *firewall* dengan berbagai macam sistem operasi Unix® yang *opensource* pada sebuah PC biasa. Sistem operasi *opensource* ini sudah banyak mendukung fungsi-fungsi *network*, termasuk *firewall*, yang tidak kalah dengan fungsi-fungsi pada produk komersial. Seiring dengan semakin banyaknya dukungan dari sistem operasi yang *opensource* untuk kebutuhan *firewall* maka saat ini juga sudah tersedia sebuah *Embedded* PC yang dijual dengan harga murah. Selain bisa dipakai sendiri, tidak menutup kemungkinan *firewall* pada *Embedded* PC juga bisa dijual seperti layaknya produk komersial meskipun fiturnya masih jauh tertinggal.

Dalam tugas akhir ini akan membahas tentang pembuatan prototipe *firewall* yang dijalankan pada sebuah *Embedded* PC yang diberi nama *GENI Firewall*. Pembuatan *firewall* ini menggunakan sistem operasi FreeBSD dengan menggunakan metodologi *non-statefull behavior*. Sistem operasi FreeBSD harus dijalankan pada RAM-Disk karena *disk space* pada *Embedded* PC tidak begitu besar, namun hal ini tidak mengurangi performanya. Hasilnya prototipe *firewall* ini mampu menggantikan *firewall* yang dibangun pada PC biasa, dengan spesifikasi *hardware* yang jauh lebih mahal, dan bahkan mencapai hasil *transfer rate* dan *boottime* yang lebih baik.

Kata kunci: *Firewall*, *Embedded* PC.

KATA PENGANTAR

Setelah mencoba dengan berbagai macam cara, akhirnya penulis dapat menyelesaikan Tugas Akhir ini dengan judul:

"Pembuatan Prototipe Firewall pada Embedded PC

dengan Menggunakan FreeBSD"

Tugas akhir ini disusun guna memenuhi salah satu persyaratan akademis untuk mendapatkan gelar Sarjana Komputer pada jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Surabaya, Januari 2004

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR GRAFIK

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pada 2 tahun terakhir ini kebutuhan akses internet khususnya untuk memenuhi kebutuhan internet di tiap daerah sangat banyak diminati. Makin banyaknya pengguna internet di seluruh Indonesia memacu penyedia jasa layanan internet atau ISP untuk semakin bagus dalam melayani pelanggannya.

Arus transfer data setiap hari untuk pengguna internet di tiap daerah bisa dibilang cukup banyak, belum lagi untuk pemakaian jalur data internal yang menghubungkan tiap titik pada satu kota. Dengan semakin banyaknya ISP di berbagai daerah, maka jaringan internet di Indonesia akan semakin kompleks.

Perusahaan ISP akan saling bersaing dalam memberikan kualitas pelayanan dan kecepatan akses data serta keamanan. Tuntutan berlangganan internet sangat dibutuhkan oleh suatu institusi untuk *go public* dan menaikkan kualitas pelayanan terhadap publik. Tapi meskipun *go public* tentunya setiap institusi mempunyai data *private* yang tidak untuk diketahui publik.

Dari sekian banyak institusi, terdapat banyak perusahaan yang akan atau sudah *go public* dengan mengandalkan internet sebagai media penyebaran informasinya. Oleh karena itu kebutuhan terhadap keamanan data sangat vital dan mendesak setiap institusi yang sudah terhubung ke internet. Kebutuhan inilah yang bisa ditangkap sebagai peluang untuk mengembangkan suatu produk yang bisa menjadi solusi terhadap permasalah itu.

Produk *firewall*, dalam bentuk *hardware-set*, yang bisa diandalkan untuk menjaga keamanan suatu *site* hampir keseluruhan di-*supply* dari luar negeri. Inilah yang menjadi tantangan untuk membuat suatu produk yang tidak kalah secara kualitas dan layak untuk dikonsumsi oleh *end user*. Minimal produk ini secara *software*—perlu diketahui jika *hardware* PC belum bisa diproduksi lokal—sudah bisa didistribusikan secara mandiri dan terpelihara dengan baik.

1.2 Permasalahan

Permasalahan yang diangkat dalam tugas akhir ini adalah :

- Desain jaringan komputer yang banyak dipakai namun ternyata kurang aman, mengakibatkan potensi untuk terjadi kerusakan sistem.
- Hal-hal yang sering muncul akibat dari efek desain jaringan komputer yang kurang aman.
- Bagaimana membangun sistem jaringan komputer yang aman dan implementasi dari *firewall*.
- Membuat pengertian *firewall* agar lebih mudah dimengerti dan dipahami.
- Apa yang diperlukan untuk membangun *firewall* dan dengan sistem operasi apa yang mungkin untuk dibuat sebagai *firewall*.
- Bagaimana mengimplementasi sistem operasi FreeBSD yang mampu berfungsi sebagai *firewall* dan bagaimana konfigurasinya.
- Arsitektur *firewall* yang dibangun pada sebuah *Embedded* PC dan bagaimana cara membuatnya.

- Konstruksi secara keseluruhan agar perangkat *firewall* ini berpotensi menjadi produk massal dan layak jual.

1.3 Tujuan

Tujuan dari tugas akhir ini adalah memberikan hasil studi tentang sistem keamanan jaringan internet dengan menggunakan firewall. Hasil studi ini akan ditunjang dengan implementasi ke sebuah alat, yaitu mengimplementasikan sistem operasi FreeBSD pada sebuah *Embedded* PC yang bisa berfungsi sebagai firewall dan mudah untuk dioperasikan oleh pengguna. Serta beberapa fitur tambahan yang akan ditambahkan agar alat ini nantinya sangat berpotensi menjadi produk massal yang layak jual.

1.4 Batasan Masalah

Dari permasalahan-permasalahan di atas, maka batasan dalam tugas akhir ini adalah:

- Studi mengenai sistem keamanan jaringan ini terbatas pada *firewall*, dan lebih banyak membahas permasalahan spesifik dan teknis daripada permasalahan sistem keamanan komputer secara keseluruhan.
- Implementasi *firewall* ini hanya mendukung untuk keluarga protokol TCP/IP versi 4, sedangkan mengenai kriptografi dan protokol lainnya tidak didukung dalam implementasinya.
- Perangkat yang digunakan adalah *Embedded* PC dengan arsitektur dasar i386 dengan media penyimpanan *compact flash* 16Mbyte.

- Dalam kasus ini tidak dibahas secara mendalam tentang sistem operasi FreeBSD, optimasi pada sistem operasi FreeBSD, serta algoritma *network code* dalam sistem operasi FreeBSD mengenai dukungan *firewall*.
- *Testing* yang akan dilakukan untuk menguji hasil dari alat ini tidak akan dibandingkan dengan produk komersial yang lain karena keterbatasan perangkat, namun hanya dibandingkan dengan produk *hand-made* yang dibangun di sebuah *Desktop* PC atau PC biasa.

1.5 Metodologi Pembuatan Tugas Akhir

Pembuatan tugas akhir ini terbagi menjadi beberapa tahapan sebagai berikut :

- Studi Literatur dan Pengumpulan Data

Tahap ini merupakan tahap pengumpulan informasi yang diperlukan untuk melakukan studi sistem keamanan jaringan. Diperoleh dengan membaca literatur-literatur yang berhubungan dengan keamanan jaringan, sistem operasi FreeBSD, dan teknologi *firewall*.

- Perancangan Sistem

Tahap ini merupakan perancangan sistem yang akan dikembangkan dan kebutuhan apa saja yang diperlukan. Termasuk memperhitungkan siapa saja yang membutuhkan sistem ini dan seberapa penting.

- Desain Sistem

Tahap ini akan menghasilkan sebuah prototipe produk seperti yang telah direncanakan pada tahap sebelumnya.

- Konstruksi Sistem

Dalam tahap ini dilakukan pembuatan sistem sampai selesai dari hasil prototipe awal yang telah dibuat. Semua ini menggunakan sistem operasi FreeBSD beserta *software* tambahan sebagai pendukungnya: php, sh script, dll.

- Implementasi dan Uji Coba

Dalam tahap ini dilakukan pengujian terhadap implementasi alat yang telah selesai dikembangkan dengan mengikuti skenario yang sudah ditentukan. Dan melakukan pembenahan-pembenahan apabila ditemukan suatu kesalahan pada saat dilakukan pengujian.

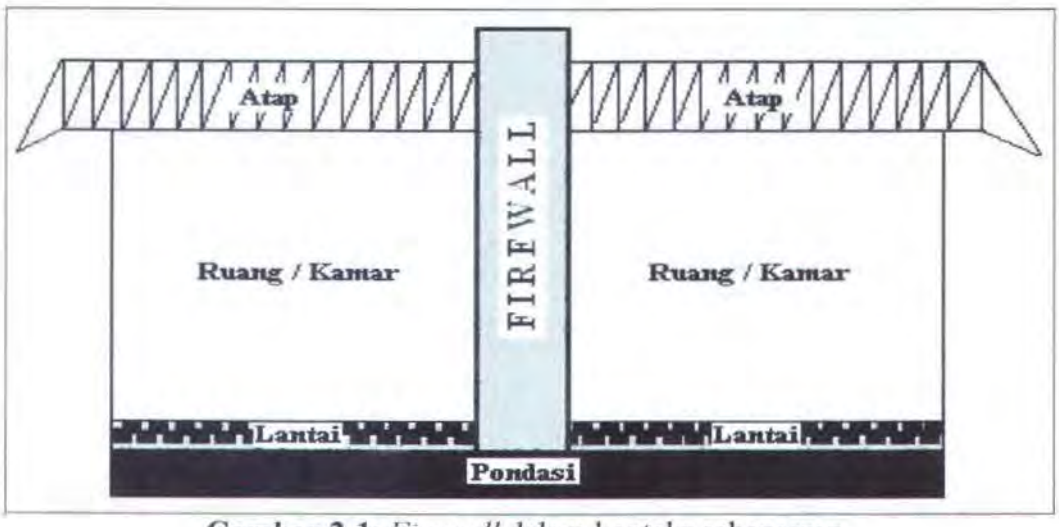
- Penyusunan Buku Tugas Akhir

Tahap ini merupakan tahap akhir, yaitu penyusunan buku sebagai laporan yang diwujudkan dalam bentuk buku Tugas Akhir.

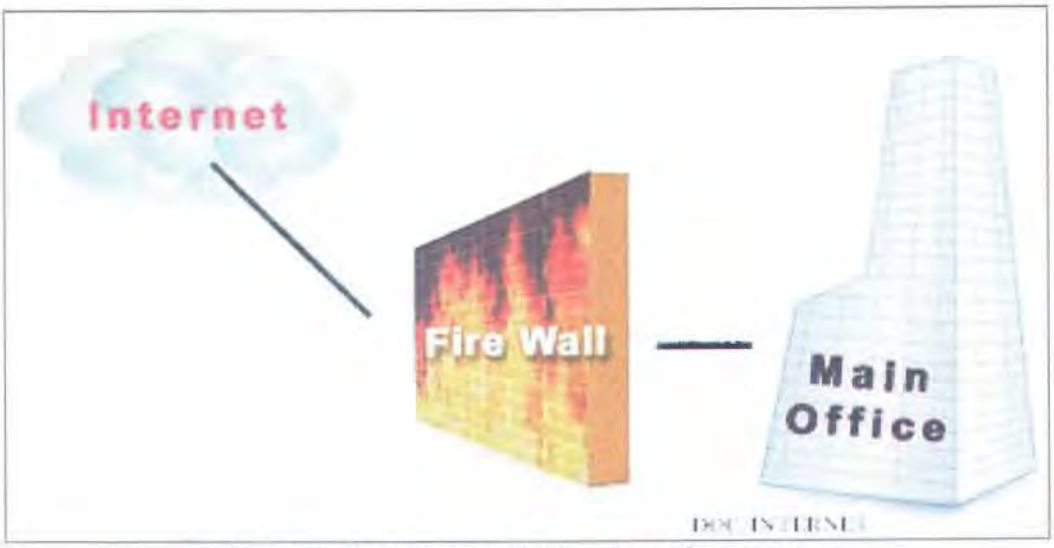
BAB II
TEORI PENUNJANG

2.1 Memahami Internet Firewall

Dalam sejarahnya, *firewall* adalah suatu konstruksi bangunan yang didesain untuk menahan penyebaran api dari satu ruangan ke ruangan yang lain, seperti ditunjukkan dalam ilustrasi **gambar 2-1**. Dalam disiplin ilmu komputer, khususnya jaringan komputer, istilah *firewall* digunakan untuk suatu perangkat yang berfungsi menahan bahaya internet menuju ke jaringan internal kita, seperti ditunjukkan dalam **gambar 2-2**. Secara teori, *firewall* dalam sebuah jaringan komputer mempunyai tujuan yang sama halnya dengan *firewall* dalam konteks sebenarnya. Seperti kita ketahui, bahaya yang datang dari internet sangat banyak dan beragam, kita ibaratkan seperti api, yang sewaktu-waktu bisa mengancam keamanan jaringan internal kita. *Firewall* sebagai langkah pencegahan yang handal, bisa dipakai pada bermacam topologi jaringan baik untuk internet maupun intranet.

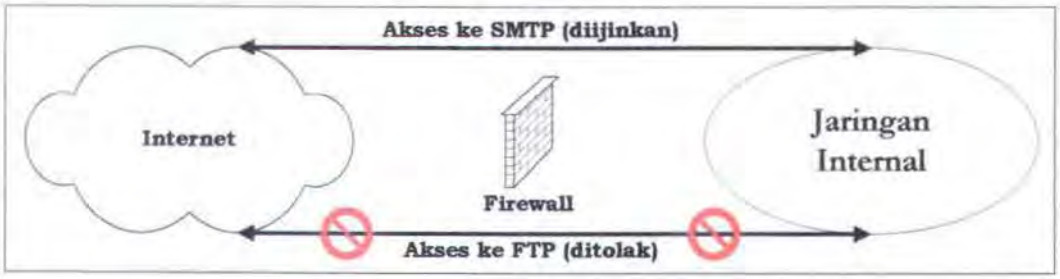


Gambar 2-1: *Firewall* dalam konteks sebenarnya.



Gambar 2-2: *Firewall* dalam konteks Internet.

Menurut buku "Building Internet Firewalls" ([CZ95]) yang diterbitkan oleh O`Reilly, *firewall* adalah suatu komponen atau kumpulan komponen yang membatasi akses antara sebuah intranet (jaringan internal) dengan internet, atau yang membatasi antara satu intranet dengan intranet yang lain. *Firewall* dimaksudkan untuk membatasi orang yang boleh masuk dan melakukan aktivitas pada *site* kita serta mencegah penyerang untuk mendekati atau merusak. Semua trafik yang datang atau keluar dari intranet kita dilewatkan melalui *firewall*. Oleh karena itu, *firewall* akan memastikan apakah trafik tersebut diterima atau tidak. Trafik akan diterima apabila trafik tersebut adalah trafik yang memang kita ijinkan untuk lewat dan tidak kita anggap berbahaya, seperti ditunjukkan pada **gambar 2-3** yang merupakan contoh pembatasan terhadap trafik yang diijinkan untuk lewat. Akses ke *file transfer* ditolak untuk lewat namun akses ke SMTP diijinkan.



Gambar 2-3: Pembatasan trafik yang boleh lewat.

Secara logika, *firewall* adalah suatu pemisah, pembatas, atau suatu *analizer* terhadap trafik yang lewat. Implementasi *firewall* secara fisik bisa berupa satu atau banyak komponen *hardware*, *router*, komputer, kombinasi beberapa komputer, kombinasi beberapa *router*, atau sebuah jaringan dengan aplikasi software pendukungnya. Banyak cara untuk mengkonfigurasi perangkat tersebut, bergantung pada *security policy* (kebijakan sistem keamanan) pada *site* kita, biaya yang kita punya, dan operasional jaringan secara keseluruhan.

2.2 Mengapa Membutuhkan Firewall

Semakin banyak perusahaan atau institusi yang melangkah *go public* demi sebuah kemajuan. Satu hal yang hampir pasti adalah mereka menggunakan internet sebagai salah satu media terpenting untuk berkomunikasi dan mempublikasikan diri. Seiring dengan itu pula banyak perusahaan atau institusi yang merasa khawatir terhadap serangan dari orang tidak bertanggung jawab lewat media internet. Dalam kurun waktu terakhir, serangan lewat internet semakin berbahaya dan semakin kompleks baik secara teknis maupun mentalitas.

Firewall, adalah salah satu cara terbaik untuk mencegah serangan lewat internet yang semakin kompleks dan berbahaya. Suatu serangan yang cukup

dahsyat bisa mengakibatkan kerusakan sistem secara fatal dan ini harus dihindari sedini mungkin. Untuk menghindarinya maka sistem harus dilindungi dan dijaga agar sewaktu-waktu tidak terkena serangan, untuk itulah *firewall* sangat dibutuhkan. Khususnya di Indonesia, fenomena mengenai keamanan jaringan sudah mulai direspon oleh pemerintah dengan dibentuknya divisi *cyber crime* oleh aparat kepolisian.

Dengan semakin merebaknya "kejahatan" di internet dan untuk melengkapi pencegahan serangan secara dini, maka disarankan kepada suatu institusi untuk menentukan suatu *security policy* (kebijakan sistem keamanan) dalam membangun sekaligus memelihara sistem jaringannya. Untuk alasan itu pula sangat direkomendasikan pada suatu institusi yang akan merencanakan kebijakan sistem keamanannya untuk menggunakan *firewall*.

2.3 Perlindungan Terhadap Resources

Terhubung dengan internet memang ada keuntungan dan kerugiannya. Kerugiannya adalah hal-hal yang selama ini dijaga sebagai informasi konfidensial akan sangat berpotensi untuk ikut terpublikasi lewat internet baik disengaja maupun tidak. Namun keuntungannya pun tidak kalah banyak, yang jelas sarana untuk membuat institusi menjadi *go public* lewat media internet sudah bisa tercapai dan peluang untuk bisa dikenal di seluruh dunia sangat besar.

Firewall adalah suatu komponen pengaman jaringan komputer. Tetapi secara lebih luas *firewall* akan menjadi salah satu komponen penting untuk melindungi sistem terkomputerisasi dalam sebuah jaringan terintegrasi. Jadi ada

beberapa hal yang harus kita lindungi agar sistem tidak menjadi korban serangan, yaitu:

- Data, yaitu suatu informasi yang kita simpan di dalam komputer dan perlu kita jaga kerahasiaannya. Tidak seorangpun boleh melihat apalagi merubah data tersebut kecuali hanya beberapa orang yang sudah dilegitimasi.
- Komputer itu sendiri, yaitu sumber daya terpenting (*critical resources*) yang sehari-hari kita gunakan untuk bekerja secara *online*. Akan sangat menyusahkan apabila setiap beberapa hari sekali kita harus melakukan instalasi ulang pada *software*-nya karena kerusakan akibat gangguan pada sistem kita.
- Reputasi perusahaan atau institusi. Perusahaan atau institusi tentu sangat menjaga nama baiknya di mata publik. Reputasi inilah yang menjadi sasaran penyerang untuk dijatuhkan. Penyerang akan melakukan penyusupan dan memalsukan identitasnya sebagai orang yang cukup penting di suatu perusahaan atau institusi. Contohnya: Perusahaan "X" harus menghadapi tuntutan dari mitra bisnisnya gara-gara sebuah email yang berisi pemutusan kontrak secara sepihak, padahal email tersebut merupakan ulah penyusup yang menyamar sebagai direktur perusahaan "X".

2.4 Macam Serangan

Ada banyak macam serangan yang mungkin terjadi pada suatu jaringan komputer. Menurut buku "Building Internet Firewalls" ([CZ95]), serangan dapat dibedakan menjadi tiga kategori dasar sbb:

- Penyusupan atau *intrusion*. Kebanyakan serangan yang mungkin dan akan terjadi pada sistem kita adalah jenis ini. Dengan cara ini, seorang penyusup akan bisa dengan mudah menggunakan komputer kita dan menyamar sebagai orang yang merupakan anggota yang sudah terlegitimasi pada suatu institusi.
- *Denial of Service*. Serangan ini bekerja untuk mematikan *service* di server kita yang sedang aktif. Misalnya, kita membuka *service* telnet di server yang bisa diakses dari mana saja. Untuk melakukan *denial of service* maka seorang penyerang akan membuat suatu program untuk melakukan *request* ke server telnet kita sebanyak 1024 koneksi secara bersamaan. *Request* yang dijalankan oleh penyerang tadi akan membuat *service* telnet di server kita mati (shutdown) dengan sendirinya karena kemampuan koneksi yang mampu dibuat oleh server telnet hanya 256 koneksi dalam waktu bersamaan.
- Pencurian informasi atau *information theft*. Seperti beberapa kategori diatas, tujuan dari beberapa penyusup adalah untuk mendapatkan sesuatu yang mereka cari dengan memanfaatkan kelemahan dari sistem. Secara teknis, biasanya mereka melakukan *packet sniffing* atau *exploit* pada sistem kita yang lemah agar bisa melakukan akses pada level yang seharusnya tertutup untuk orang luar. Selain itu bisa juga dengan cara *social engineering*, yakni dengan mempelajari hal-hal diluar teknis untuk mengorek kelemahan sistem kita. Misalnya untuk mengetahui *password* sistem dari *administrator*, maka dipelajari latar belakang dan kehidupan sehari-hari *administator* –tanggal lahir, nama pacar, makanan favorit, dsb– dengan harapan *password* yang digunakan adalah salah satu diantaranya.

Ketiga macam kategori dasar mengenai jenis serangan bisa terjadi karena banyak sebab dan tujuan. Sebab pertama adalah hal teknis, yaitu suatu kelemahan dari sistem jaringan yang kita bangun sehingga membuka peluang untuk diserang. Dan sebab kedua merupakan kecerobohan dari *administrator* atau penggunanya. Kelemahan *administrator* dalam mendesain jaringannya merupakan faktor terpenting. *Administrator* jaringan harus mengetahui bagaimana membangun jaringan yang baik dan aman. *Firewall* tidak didesain untuk mencegah kerusakan yang disebabkan karena kecerobohan dan kesalahan penggunaan.

Selain kelemahan dari *administrator* adalah kurang mengertinya pengguna dalam mengoperasikan sistem yang ada. Sebuah studi mengestimasi bahwa 55% dari penyebab terjadinya kerusakan sistem adalah ketidaktahuan pengguna ([CZ95]). Jadi selain melindungi sistem dari serangan oleh orang luar maupun dalam, adalah memberikan *user-education* untuk mengeliminasi terjadinya kerusakan pada sistem kita. Seperti banyak disebutkan dalam beberapa referensi bahwa keamanan jaringan lebih banyak merupakan masalah mentalitas daripada masalah teknis.

2.5 Karakteristik Penyerang

Beberapa macam penyerang atau orang usil yang mengganggu sistem kita mempunyai beberapa karakteristik yang unik. Tidak semuanya merupakan tindak kejahatan penjahat. Pada kenyataannya banyak macam serangan yang mungkin terjadi pada jaringan kita, dan para penyerang tersebut mempunyai karakteristik

yang berbeda-beda. Menurut buku "Building Internet Firewalls" ([CZ95]), ada empat karakteristik dasar sbb:

- *Joyriders* atau mungkin dalam bahasa kita adalah orang yang suka dengan petualangan dan hal baru. Orang ini mancari kesenangan dengan mengganggu *site* orang lain karena merasa bosan dengan aktivitasnya. Mereka adalah orang yang rasa ingin tahunya tinggi, dan ingin menunjukkan kebolehan atau keahliannya dalam bidang komputer.
- *Vandals* atau perusak. Penyerang jenis ini sangat mengganggu dan bisa dikatakan sebagai musuh bagi perusahaan atau institusi yang menjadi tempat bekerja bagi orang banyak. Karena dengan dirusaknya sistem, maka akan mengakibatkan pekerjaan orang banyak akan terganggu. Mereka merusak dengan beberapa alasan: ada yang karena benci atau balas dendam, ada pula yang memang gemar merusak dan merupakan suatu kepuasan tersendiri bila telah berhasil merusak.
- *Score keeper*. Agak sulit untuk menerjemahkan langsung ke bahasa kita. Karakteristik penyerang ini bertujuan untuk mematahkan rekor suatu sistem yang sudah dikenal bagus dan aman. Mereka akan mengoleksi *site* yang berhasil mereka serang, semakin banyak *site* yang "dikuasai" akan menunjukkan kualitas mereka. Meski mereka umumnya tidak merusak, namun akan memasang suatu *backdoor* (lubang yang sengaja ditinggalkan agar sewaktu-waktu bisa digunakan) pada *site* tersebut. Biasanya *backdoor* ini untuk saling ditukar dengan sesama penyerang, atau *backdoor* dipergunakan untuk menyerang *site* lain. Jadi jangan kaget misalnya tiba-tiba kita dituduh

menyerang suatu *site* karena ternyata *site* kita dipergunakan oleh *score keeper* untuk menyerang *site* lain.

- *Spies* atau mata-mata, bisa karena persaingan antar perusahaan atau yang lain. Penyerang jenis ini melakukan penyusupan dengan sangat rapi, hanya untuk mencuri informasi yang menurut mereka bisa menjadi "kartu as" untuk menjatuhkan lawan. Namun pada beberapa kasus bisa jadi *spies* akan menjual informasi tersebut meski mereka bukan bekerja sebagai mata-mata secara profesional.

Setelah mengetahui karakteristik penyerang, tentu kita harus senantiasa waspada. Para penyerang bisa berasal dari mana saja dan lewat mana saja. Jalur internet menjadi pilihan efektif bagi penyerang karena jauhnya jarak yang harus mereka tempuh untuk datang ke tempat kita. Ada juga cara lain yaitu datang ke tempat kita (mungkin dengan cara menyamar) kemudian mencari titik lemah sistem milik kita yang potensial untuk dibobol. Mereka mungkin adalah orang yang tidak dikenal sama sekali dan berada jauh dari tempat kita, namun bisa jadi malah sebaliknya.

2.6 Kelebihan dan Kekurangan Firewall

Seperti dalam falsafah Tzun-Zhu, "Menghadapi pasukan yang terampil dalam menyerang, musuh tidak tahu bagaimana harus bertahan. Sedangkan menghadapi pasukan yang terampil dalam bertahan, musuh tidak tahu bagaimana harus menyerang". Sama halnya dengan *firewall*, namun disini *firewall* bukan suatu alat untuk menyerang melainkan suatu sistem pertahanan yang handal. "Jika

musuh menyerang dengan cara lama, kita kalahkan dengan cara lama pula", kata pepatah kuno. Pada kenyataannya, *firewall* akan dapat melakukan hal-hal sbb:

- Menjadi tumpuan untuk memfokuskan suatu sistem keamanan jaringan. Logikanya, *firewall* adalah suatu ujung dari batas wilayah jaringan kita yang terhubung ke internet. Semua trafik harus melalui *firewall* baik ke dalam maupun ke luar. Akan lebih mudah apabila kita memfokuskan sistem keamanan kita pada satu titik daripada menyebarkan beberapa teknologi keamanan jaringan pada satu wilayah kita sekaligus.
- Merupakan perwujudan dari kebijakan sistem keamanan suatu jaringan. Misalnya kita hanya memperbolehkan akses *file transfer* (FTP *Service*) dari internet ke salah satu server kita, maka *firewall* bisa menjadi *watchdog* (anjing penjaga) yang akan melarang atau menolak semua trafik dari internet ke *site* kita kecuali untuk *file transfer* saja.
- Mencatat trafik secara efisien. Karena semua trafik baik ke luar maupun ke dalam pasti melalui *firewall*, maka *firewall* akan menyediakan suatu tempat dan layanan untuk menyimpan aktivitas trafik serta memberikan *report* yang berguna bagi kita.
- Membatasi akses antar komputer di intranet kita. Terkadang, *firewall* menjadi pemisah antar bagian dari jaringan intranet kita. Misalnya, di suatu perusahaan ada dua jaringan yaitu jaringan untuk bagian keuangan dan bagian teknik. Dalam hal ini *firewall* akan menjadi pemisah antara kedua jaringan tersebut. Dengan begitu kebijakan keamanan jaringan adalah membatasi akses untuk masing-masing departemen.

Selain hal-hal yang bisa dilakukan oleh *firewall*, maka ada juga hal-hal yang tidak dapat dilakukan oleh *firewall*, yaitu:

- Melindungi jaringan dari orang dalam yang jahat (*malicious insider*). *Firewall* hanya akan melindungi data atau informasi yang akan dicuri oleh penyusup yang terhubung ke sistem melalui media internet atau intranet. Namun apabila ada orang dalam yang memang berniat untuk mengambil data secara ilegal dengan maksud jahat, maka *firewall* tidak dapat melindunginya. Karena bisa jadi orang dalam atau *malicious insider* tersebut melakukannya dengan cara pemaksaan secara fisik, misalnya mencuri data dengan cara membongkar komputer secara paksa. Atau memang pelakunya adalah "orang penting" yang mempunyai hak akses istimewa pada sistem kita dan tidak kita duga sama sekali.
- Melindungi suatu *site* dari suatu koneksi yang tidak melewati *firewall*. Misalnya topologi jaringan kita hanya menerapkan *firewall* untuk koneksi yang ke internet saja, padahal masih ada koneksi kita yang terhubung ke tempat lain tanpa melalui *firewall*. Maka dalam hal ini *firewall* tidak bisa melindunginya.
- Tidak dapat melindungi dari suatu serangan yang dilakukan dengan cara baru dan kompleks yang tidak dapat dikenali oleh *firewall*. contoh kasusnya adalah: suatu serangan dengan teknik DOS (*Denial of Service*) dapat ditahan karena sudah dikenal oleh *firewall*. Namun dalam seiring kemajuan teknologi, serangan dengan teknik ini berkembang menjadi teknik DDOS (*Distributed Denial of Service*). Penyerang melakukan penyerangan ke suatu *site* dengan

menggunakan "jasa" *site* lain yang sudah dikuasainya. Misalnya, *score keeper* akan menyerang *site* kita secara bersamaan dari beberapa *site* yang sudah menjadi korbannya terlebih dahulu. *Firewall* tidak dapat menahan serangan dengan teknik DDOS apabila tidak dilakukan *upgrade* terlebih dahulu.

- Melindungi sistem dari virus. *Firewall* sangat sulit untuk melindungi sistem dari serangan virus karena *firewall* sangat sulit untuk melakukan hal-hal sbb:
  . Mengenali bahwa paket data yang lewat merupakan bagian dari suatu program aplikasi.
  . Menetapkan bahwa paket data yang lewat adalah suatu karakteristik dari suatu virus.
  . Menganalisa suatu perubahan pada paket data yang lewat adalah akibat dari suatu virus.

  Membuat *firewall* yang bisa mengenali virus? Ya, ini adalah tantangan.

2.7 Komponen Dasar Firewall

Firewall mempunyai beberapa komponen dasar dan definisi yang harus diketahui sebelum membangunnya. Komponen tersebut definisikan sebagai berikut:

- *Host.* Sebuah komputer yang terhubung dalam suatu jaringan.
- *Bastion Host.* Sebuah komputer yang harus diamankan secara khusus karena sangat potensial untuk diserang atau karena merupakan pusat data elektronik yang sangat penting.

- *Dual-Homed Host.* Sebuah komputer yang terhubung dalam suatu jaringan dengan memiliki minimal 2 *interface* jaringan (dual ethernet).
- *Packet.* Suatu object yang menjadi pokok komunikasi jaringan antar komputer, sering disebut sebagai paket data.
- *Packet Filtering.* Fungsi dari suatu alat yang mengontrol aliran data dari suatu jaringan ke jaringan lainnya. Alat ini akan memperbolehkan atau menolak paket data yang lewat secara selektif. Fungsi ini sering disebut juga sebagai *screening*, dan bisa terdapat di *router*, *bridge*, atau suatu *host*.
- *Perimeter Network* atau biasa disebut DMZ. Suatu jaringan yang "diapit" antara internet dan jaringan internal untuk tujuan pengamanan berlapis.

2.8 Desain Firewall

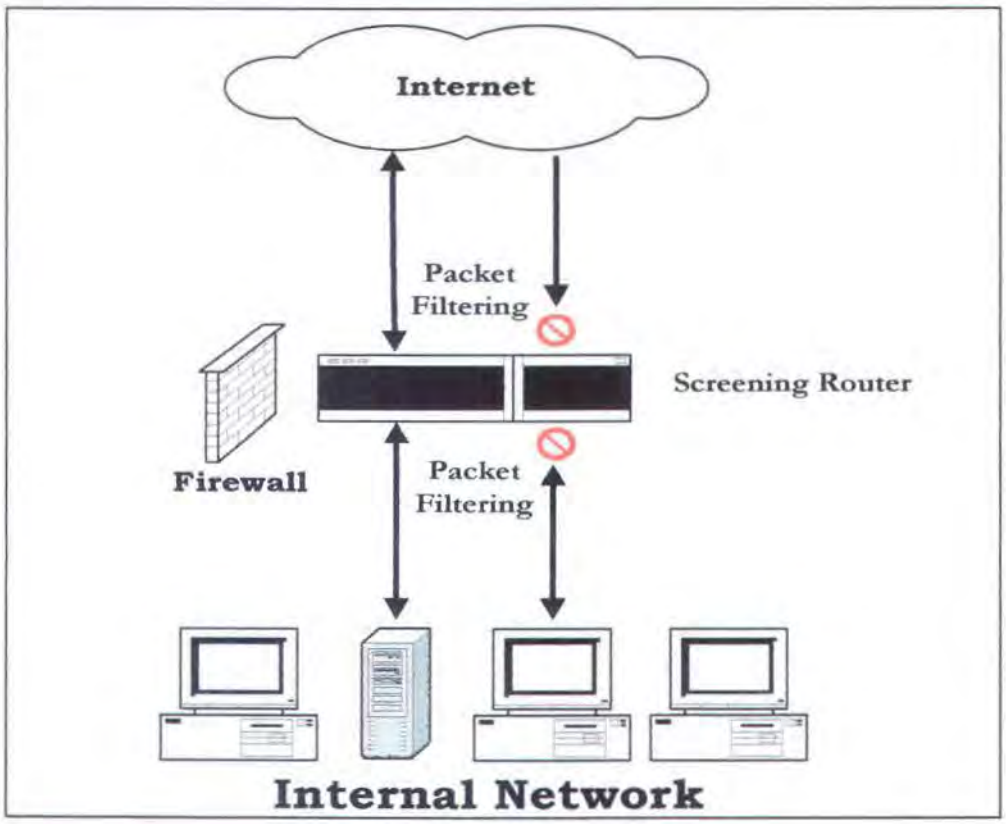
Ada tiga macam desain *firewall* ([CZ95]) yang bisa dijadikan referensi, yaitu *packet filtering*, *proxy services*, dan kombinasi *packet filtering* dan *proxy services*.

2.8.1 Packet Filtering

Firewall akan memperbolehkan dan menolak *packet* yang melewatinya sesuai dengan aturan dari *administrator*. *Router* yang digunakan untuk *firewall* ini disebut sebagai *screening router*. **Gambar 2-4** menunjukkan suatu implementasi dari *security policy* pada sebuah *site* dengan memakai desain *packet filtering*.

Desain ini mempunyai kegunaan untuk menyeleksi paket data yang lewat dari atau menuju ke jaringan internal kita:

- Memblokir semua koneksi yang berasal dari internet, kecuali akses pada protokol SMTP untuk *email.*
- Memblokir semua koneksi yang berasal dari *site* internal menuju ke sebuah alamat yang tidak diinginkan, atau juga sebaliknya.
- Memperbolehkan beberapa *service* pada jaringan internal kita terbuka, misalnya untuk akses HTTP dan FTP, tapi tidak memperbolehkan akses ke *service* selain itu.



Gambar 2-4: Desain *Packet Filtering.*

Agar *screening router* dapat berfungsi dengan baik, maka ada beberapa informasi yang harus bisa diterjemahkan oleh *router* sbb:

- IP *source address*, yaitu alamat IP dari pengirim paket data.
- IP *destination address*, yaitu alamat IP dari penerima paket data.

- Protokol, terdiri dari TCP, UDP, dan ICMP.
- TCP atau UDP *source port*, yaitu alamat *port* dari pengirim paket data.
- TCP atau UDP *destination port*, yaitu alamat *port* dari penerima paket data.

Sebagai tambahan, setiap *interface* yang terpakai harus didefinisikan agar bisa diterjemahkan oleh *screening router*, seperti dibawah ini:

- *Interface* sebagai penerima paket data.
- *Interface* sebagai pengirim paket data.

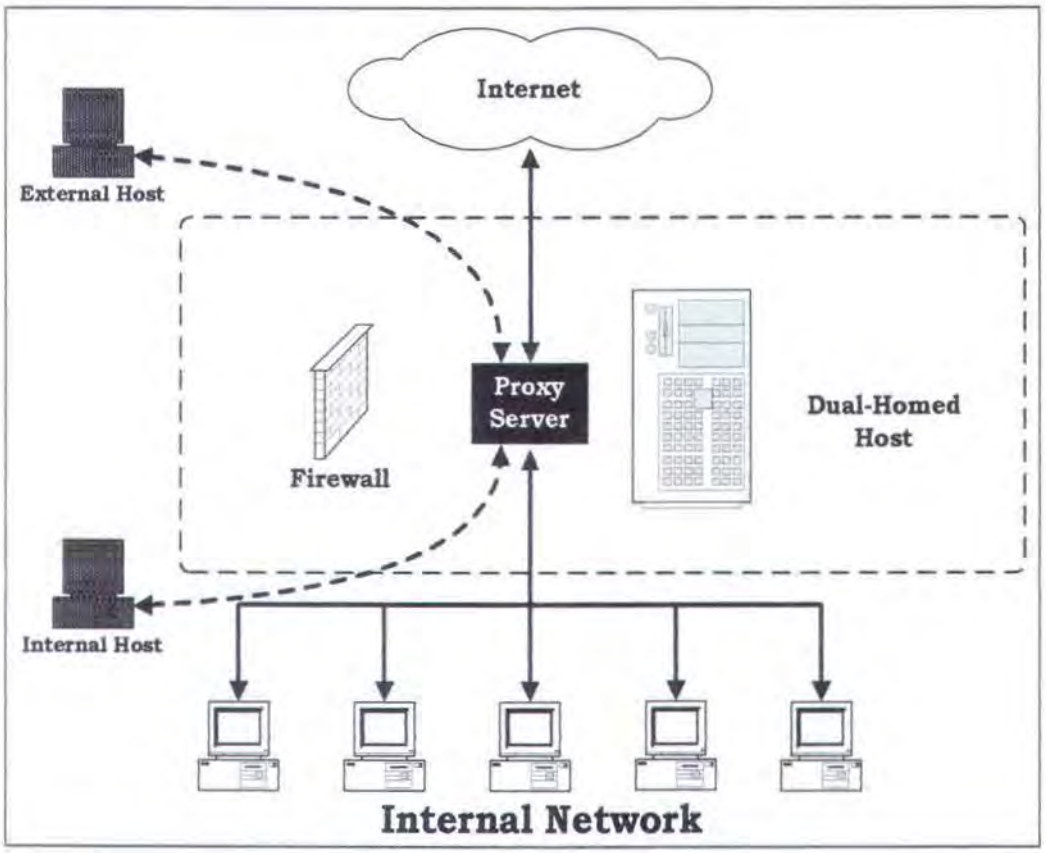
Sekilas *screening router* ini seperti router biasa, tapi sudah ditambahkan dengan fitur untuk bertindak sebagai *screener*. Dan sebagai perwujudan dari sistem keamanan jaringan, maka tanggung-jawab *screening router* jauh lebih besar daripada *router* biasa.

2.8.2 Proxy Services

Proxy services adalah sebuah program yang berjalan pada sebuah perangkat *dual-homed host* dengan salah satu *interface*-nya terhubung ke jaringan internal dan satunya lagi terhubung ke internet. Hal ini sering kita kenal dengan *proxy server*. Selain itu, *proxy service* bisa juga berjalan pada sebuah perangkat *bastian-host* yang mempunyai hak akses ke internet dan bisa diakses dari jaringan internal. *Proxy* akan berperan sebagai *gateway* dari koneksi yang sedang diminta oleh *client*, karena itu *proxy* sering disebut sebagai *application-level gateways*.

Seperti yang terlihat di **gambar 2-5** bahwa *firewall* dengan menggunakan *proxy server* secara fisik tidak jauh berbeda dengan desain sebelumnya. Hanya disini sistem keamanannya lebih difokuskan pada level aplikasi. Sehingga untuk koneksi yang harus dilakukan secara *streaming* penggunaan *proxy server* tidak

akan efektif. Karena apabila harus dibutuhkan koneksi yang bersifat *streaming*, maka koneksi dari *client* ke internet tidak membutuhkan *proxy service*.



Gambar 2-5: Desain *Proxy Services*.

Penggunaan *proxy server* merupakan solusi perangkat lunak yang memungkinkan untuk di-*customize* menurut kebutuhan. *Proxy server* dapat mengontrol apa yang dilakukan oleh *user*, karena *proxy server* dapat mengambil keputusan untuk melanjutkan atau tidak mengenai koneksi yang diminta oleh *user*. Misalnya, *proxy server* akan memblokir permintaan oleh *user* apabila pada *site* yang diakses mengandung kata-kata tidak sopan.

Ada beberapa aplikasi yang tersedia untuk membuat *proxy server*, misalnya SOCKS dan SQUID. Kedua aplikasi tersebut disediakan secara *free* dan

cukup mudah digunakan. Banyak *network administrator* yang menggunakan kedua aplikasi tersebut untuk menerapkan *firewall* pada *site*-nya.

2.8.3 Kombinasi Packet Filtering dan Proxy Services

Mungkin ini adalah "jalan tengah" untuk membuat dan mengimplementasikan *firewall* dalam sebuah kombinasi yang mengesankan. Kombinasi dua teknik desain *firewall* untuk menyelesaikan masalah yang berbeda pula. Yang harus sangat diperhatikan adalah jangan sampai kombinasi dua teknik ini justru menimbulkan masalah baru, tapi justru bisa menyelesaikan masalah yang dihadapi.

Beberapa protokol yang memang membutuhkan koneksi secara *streaming*, misalnya SMTP dan Telnet, akan di-*handle* oleh *screening router*. Sedangkan untuk protokol yang lain, misalnya FTP dan HTTP, akan di-*handle* oleh *proxy server*. Kombinasi dua teknik ini berjalan pada suatu perangkat keras yang sama, seperti pada beberapa produk *firewall* juga ada yang menerapkan cara ini.

2.9 Arsitektur Firewall

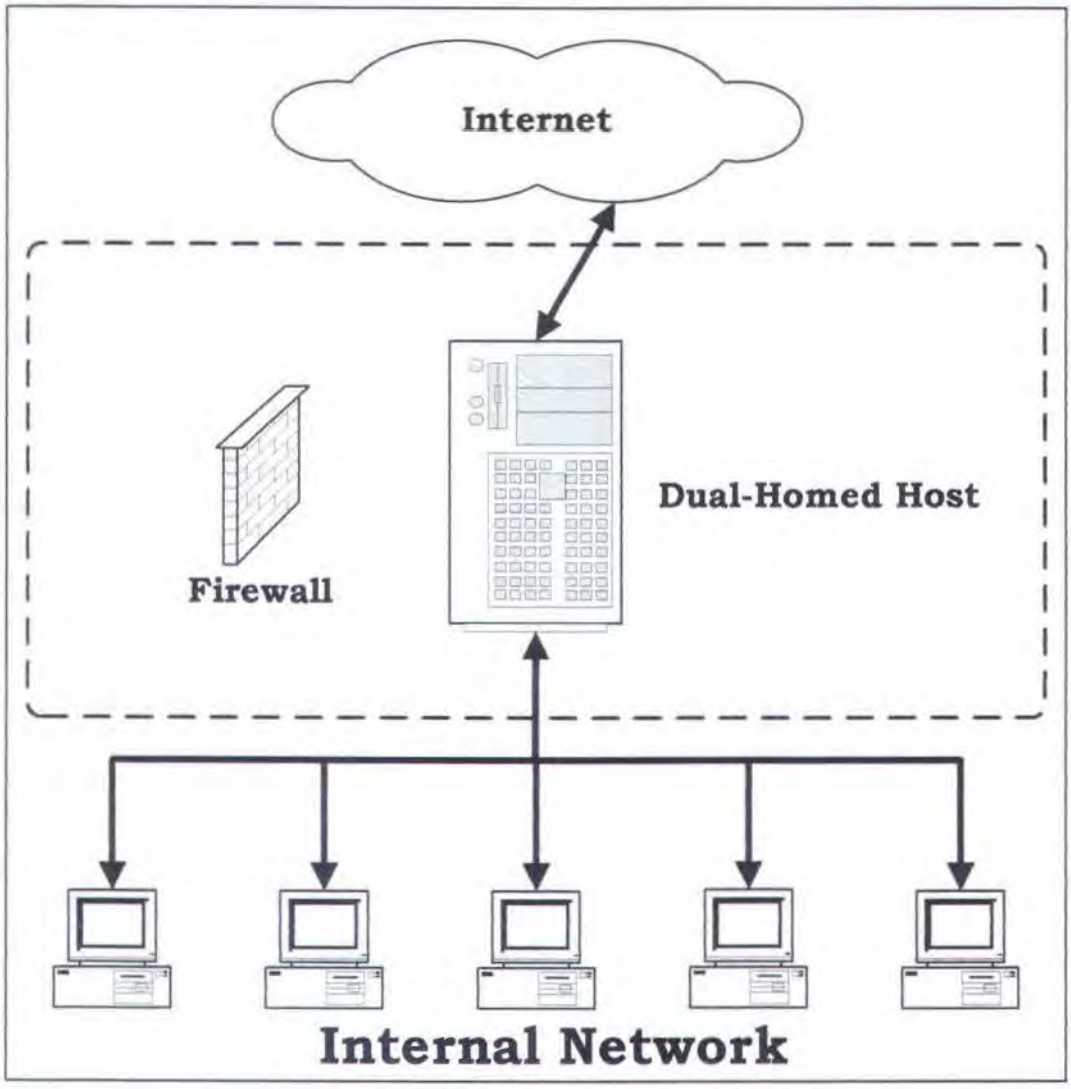
Ada tiga macam arsitektur dasar *firewall* ([CZ95]) yang bisa dijadikan referensi, yaitu *dual-homed host*, *screened host*, dan *screened subnet*.

2.9.1 Dual-Homed Host

Untuk arsitektur ini diperlukan sebuah komputer dengan minimal mempunyai dua *interface* jaringan. Dengan memakai arsitektur *dual-homed host* ini, maka tidak diperbolehkan adanya *routing* langsung antara jaringan internal dengan internet. Sistem yang ada di internet dan jaringan internal bisa

berkomunikasi secara langsung ke *dual-homed host* ini. Tapi antara jaringan internal dan internet tidak bisa berkomunikasi secara *streaming* atau langsung. Trafik untuk IP sama sekali tidak diperbolehkan.

Seperti terlihat pada **gambar 2-6** bahwa arsitektur ini dapat melakukan kontrol yang sangat ketat. Namun arsitektur ini hanya bisa melayani dengan *proxy service*. Salah satu keuntungan arsitektur ini bisa melakukan kontrol terhadap data yang diperbolehkan lewat atau tidak.

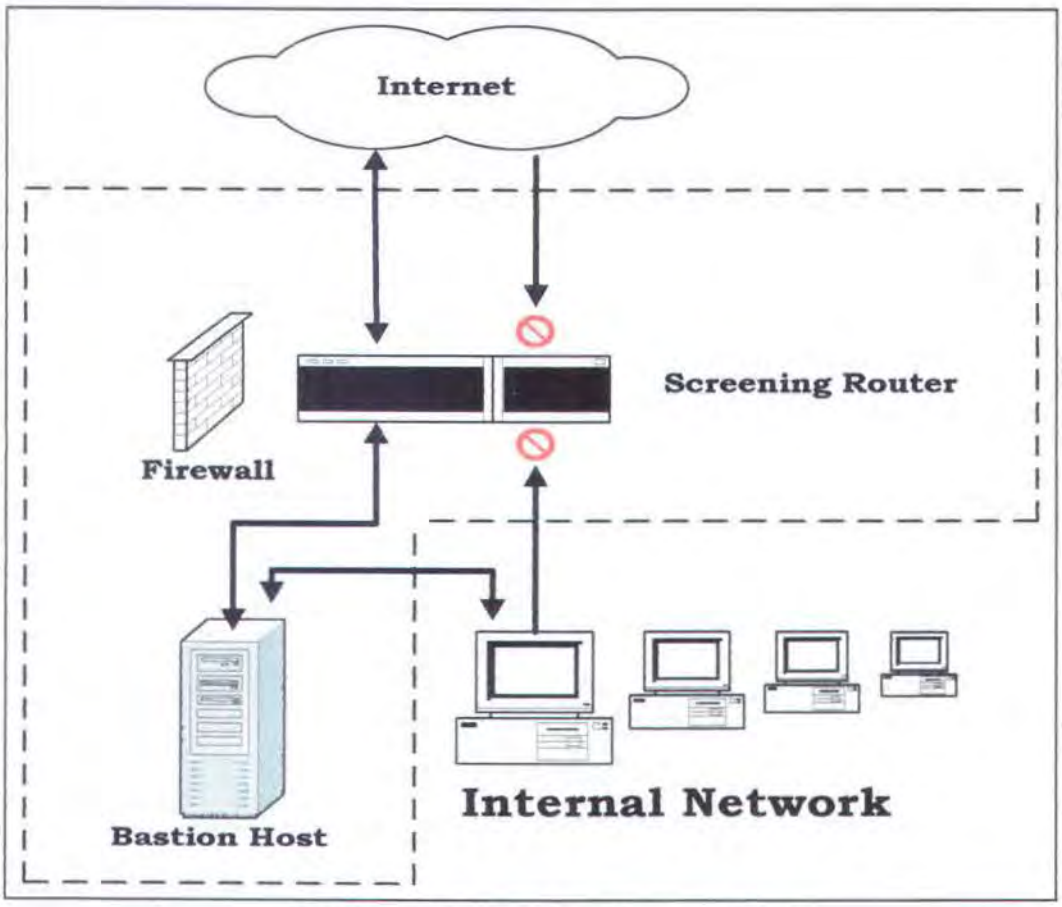


Gambar 2-6: Arsitektur *Dual-Homed Host*.

2.9.2 Screened Host

Untuk arsitektur ini sebuah *screened router* akan terhubung ke jaringan internal dan internet. Seperti pada arsitektur *dual-homed host* akan tetapi tidak mematikan fungsi *routing*. Koneksi secara *streaming* bisa dilakukan sebatas yang diijinkan saja. Arsitektur ini secara keseluruhan didukung oleh *packet filtering* yang merupakan fitur dari *screening router*.

Seperti terlihat pada **gambar 2-7** terdapat sebuah *bastion host* yang hanya diperbolehkan melakukan koneksi ke internet. Sedangkan anggota dari jaringan internal yang lain tidak diperbolehkan melakukan koneksi ke internet.



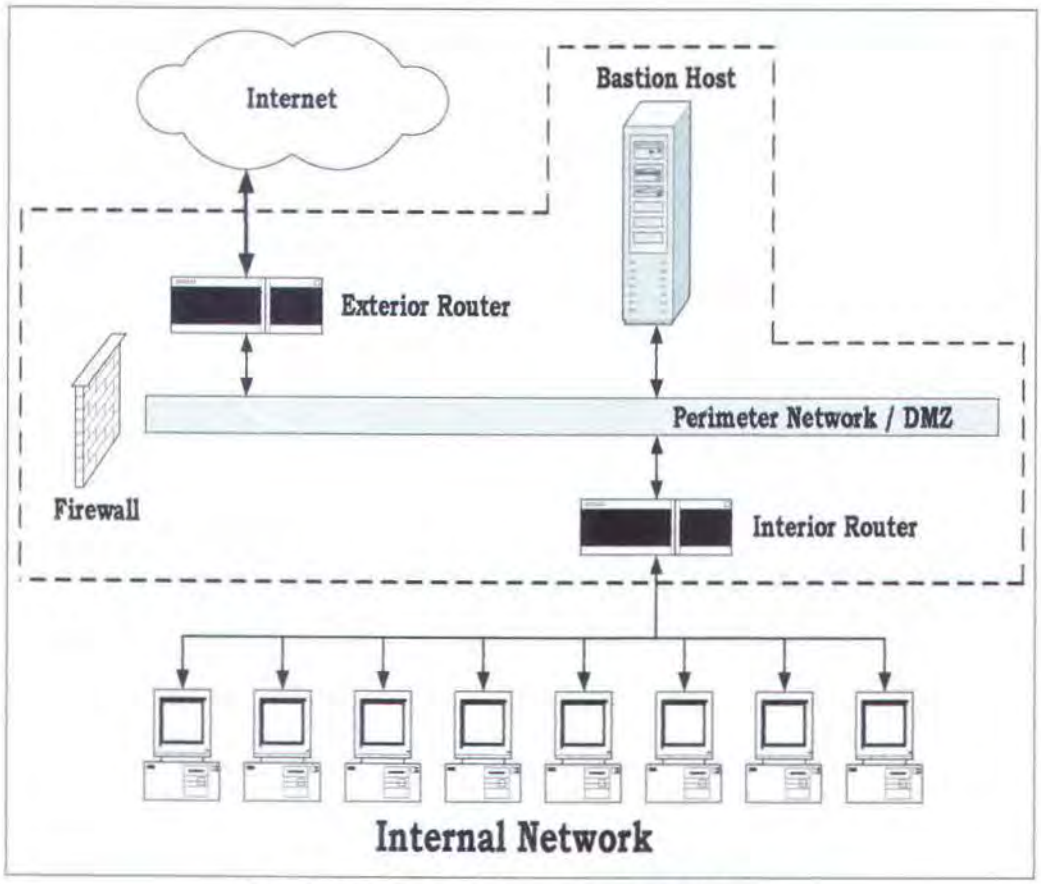
Gambar 2-7: Arsitektur *Screened Host*.

Dalam arsitektur seperti ini, biasanya *bastion host* yang dimaksud adalah sebuah *proxy server* atau *email server.* Jadi koneksi yang akan dibuat antara internet dan jaringan internal cukup aman. Sekilas arsitektur ini tampak lebih beresiko dibandingkan dengan arsitektur *dual-homed host* karena ada *routing* langsung antara jaringan internal dan internet.

Tapi meskipun demikian, dalam prakteknya, *dual-homed host* mempunyai kelemahan karena tidak bisa melakukan *packet filtering.* Banyak serangan yang tidak bisa diduga sebelumnya, dan hal ini akan membuat *dual-homed host* menjadi lemah. *Screening router* banyak menjadi pilihan karena berfungsi sangat efektif dengan kemampuan *packet filtering*-nya.

2.9.3 Screened Subnet

Arsitektur ini berbasis pada *screened host* dengan penambahkan satu lapisan jaringan yang diberi nama *perimeter network* atau DMZ. Lapisan jaringan baru atau *perimeter network* ini terisolasi dari jaringan internal dan internet, seperti terlihat pada **gambar 2-8**.



Gambar 2-8: Arsitektur *Screened Subnet.*

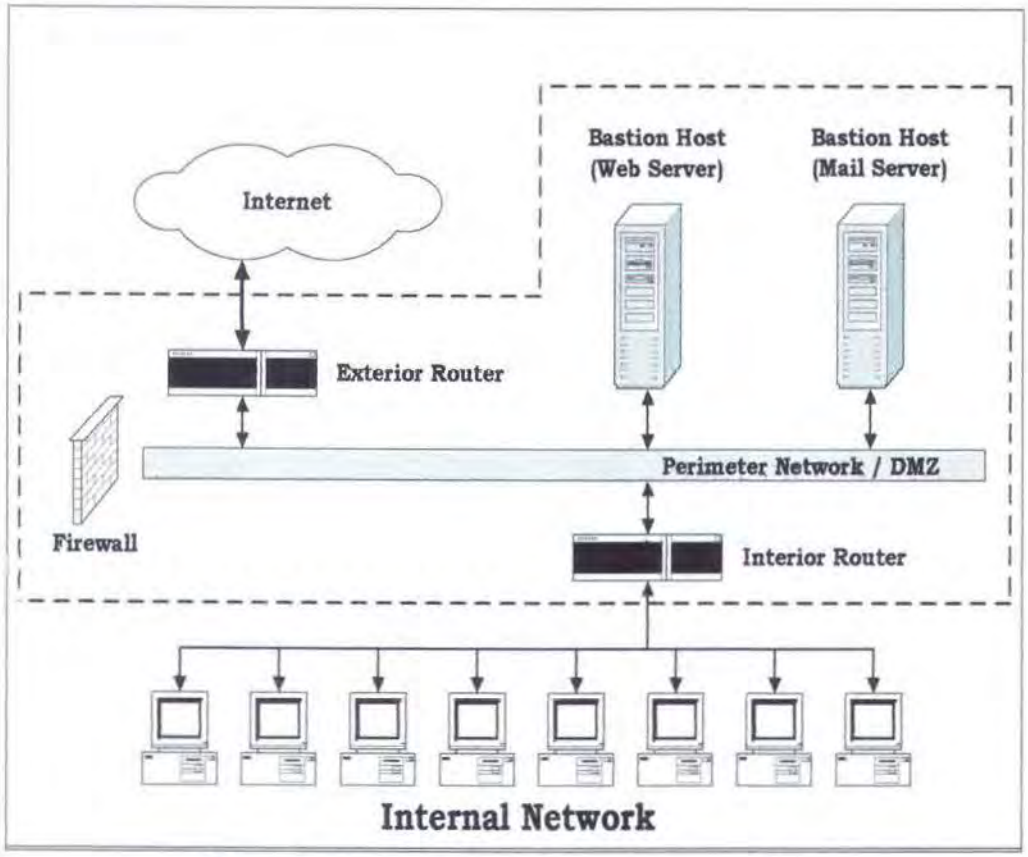
Dengan mengisolasi sebuah jaringan maka diharapkan arsitektur ini benar-benar aman dan sangat sulit dibobol. Anggota dari *perimeter network* memang dilindungi dari serangan yang berasal dari dalam maupun luar. Suatu proteksi yang sangat bagus, dengan pemikiran dasar bahwa penyerang tidak hanya muncul dari luar tapi juga dari dalam. Tipe yang cocok serta aman untuk *exterior router* dan *interior router* adalah yang mendukung *packet filtering.* Oleh karena itu pemakaian *screening router* adalah sangat tepat.

2.10 Variasi Arsitektur Firewall

Setelah mempelajari beberapa arsitektur *firewall*, maka kita akan mempelajari pula beberapa variasi dari arsitektur *firewall* ini. Ada beberapa kombinasi dan variasi yang masih bisa direkomendasikan, akan tetapi ada pula yang tidak direkomendasikan. Variasi dan kombinasi ini akan memberikan pilihan yang lebih dan penyesuaian dengan *resource* yang dimiliki. Masing-masing pasti ada keuntungan dan kerugiannya.

2.10.1 Menggunakan Banyak Bastion Host

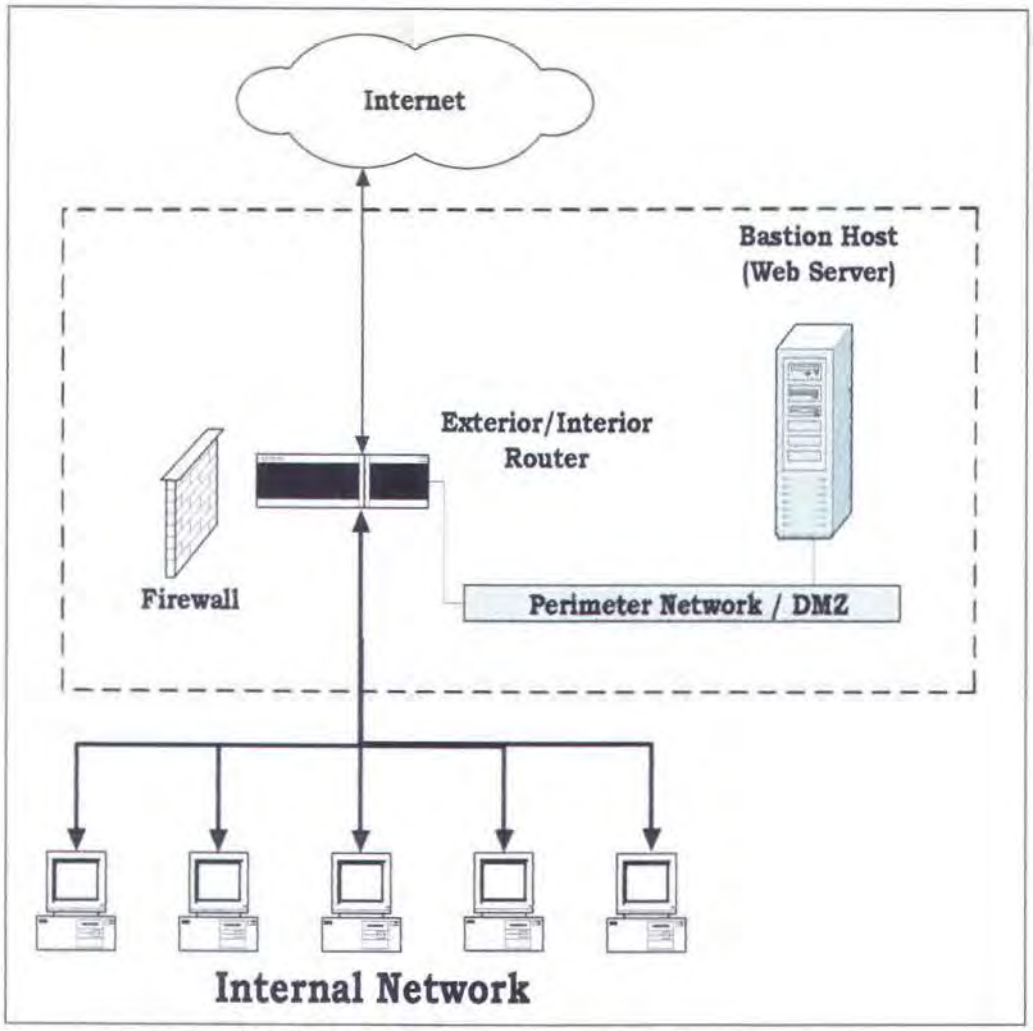
Berbasis pada arsitektur *screened subnet* yang menggunakan banyak *bastion host*. Dengan alasan performa tampaknya hal ini cukup masuk akal. Seperti terlihat pada **gambar 2-9** terjadi pemisahanan antara *host* untuk FTP/WWW dan SMTP/DNS. Dengan dua *bastion host* ini maka diharapkan layanan pada masing-masing fungsi tersebut bisa mencapai performa terbaiknya.



Gambar 2-9: Penggunaan banyak *Bastion Host.*

2.10.2 Menggabungkan Exterior dan Interior Router

Berbasis pada arsitektur *screened subnet* dengan penggabungan kedua *router* tersebut akan sangat menghemat biaya. Cukup dengan penambahan satu *interface* pada *screening router* untuk bisa menampung tiga *network* sekaligus. **Gambar 2-10** memperlihatkan sebuah penggabungan yang cukup menarik karena hanya dengan biaya yang kecil tapi bisa membuat suatu *perimeter network* dengan satu buah *router* saja.

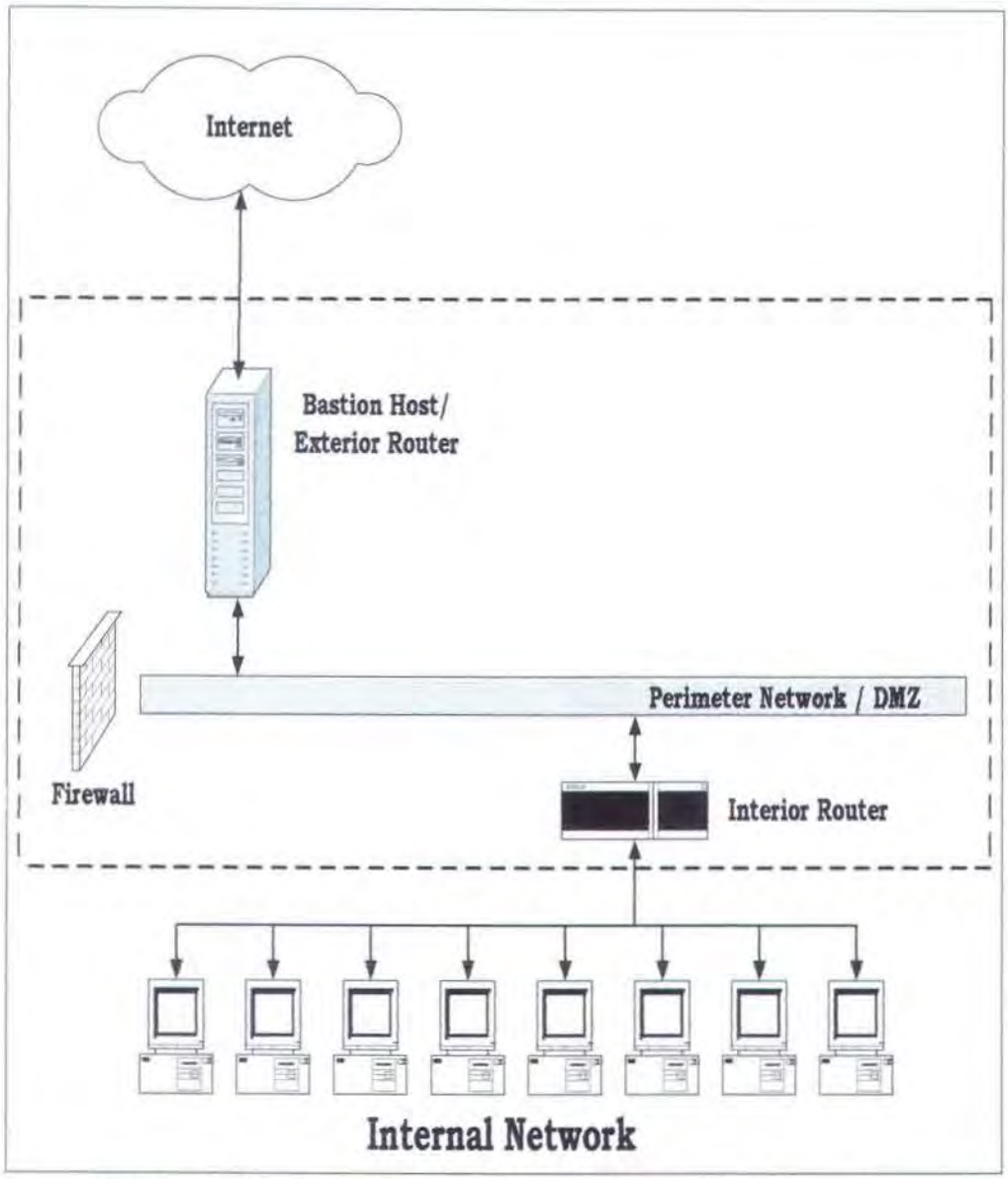


Gambar 2-10: Penggabungan *Exterior* dan *Interior Router.*

2.10.3 Menggabungkan Exterior Router dan Bastion Host

Berbasis pada arsitektur *screened subnet* dengan penggabungan *exterior router* dan *bastion host.* Arsitektur ini sangat cocok digunakan pada suatu jaringan yang tidak memerlukan koneksi secara *dedicated* atau terus-menerus akan menghemat biaya, seperti misalnya koneksi yang diinginkan adalah *dial-up* SLIP atau PPP. Maka *bastion host*, dalam hal ini adalah *dual-homed host*, sekaligus berfungsi sebagai *exterior router* yang cukup fleksibel.

Penggabungan ini tidak terlalu membuka peluang baru terhadap kelemahan sistem keamanannya. Seperti terlihat dalam **gambar 2-11** *bastion host* juga bisa difungsikan sebagai *screening router*. Namun karena hal ini belum pernah dilakukan sebelumnya, maka diperlukan sedikit ekstra konfigurasi untuk melindungi *bastion host* yang bersinggungan langsung dengan internet.

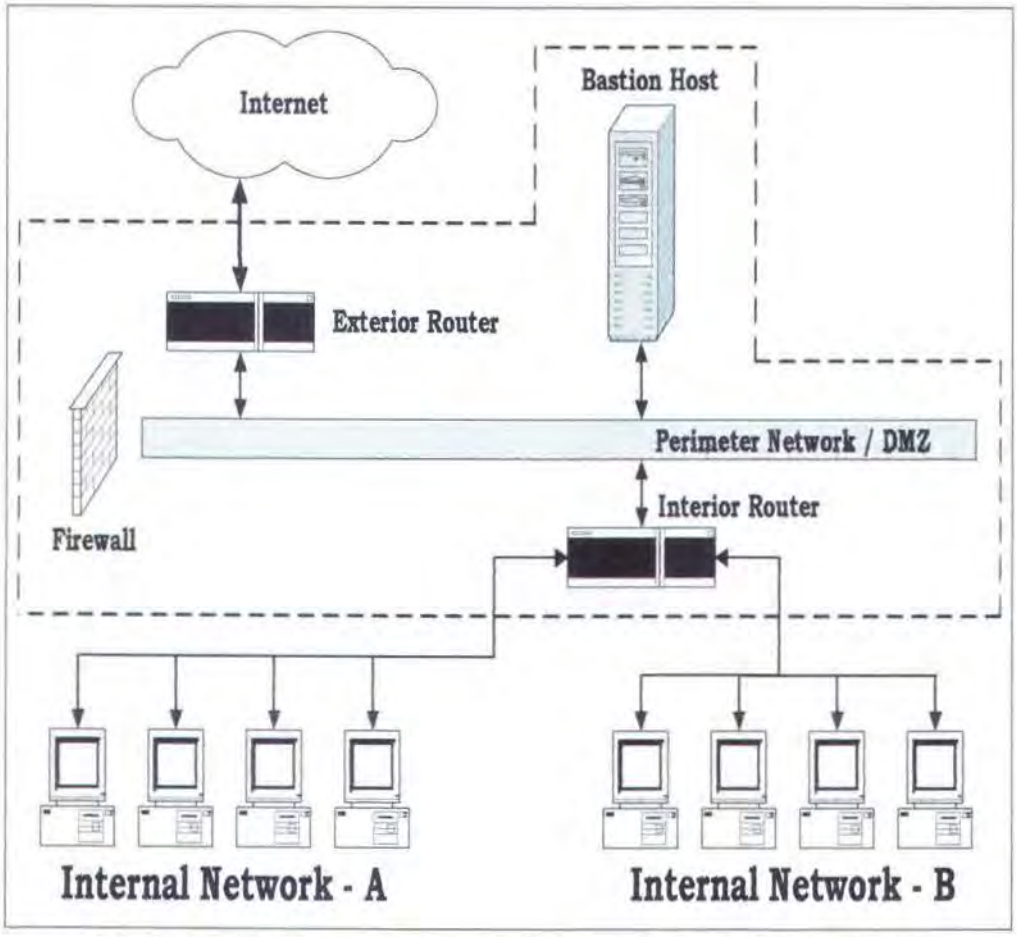


Gambar 2-11: Penggabungan *Exterior Router* dan *Bastion Host*.

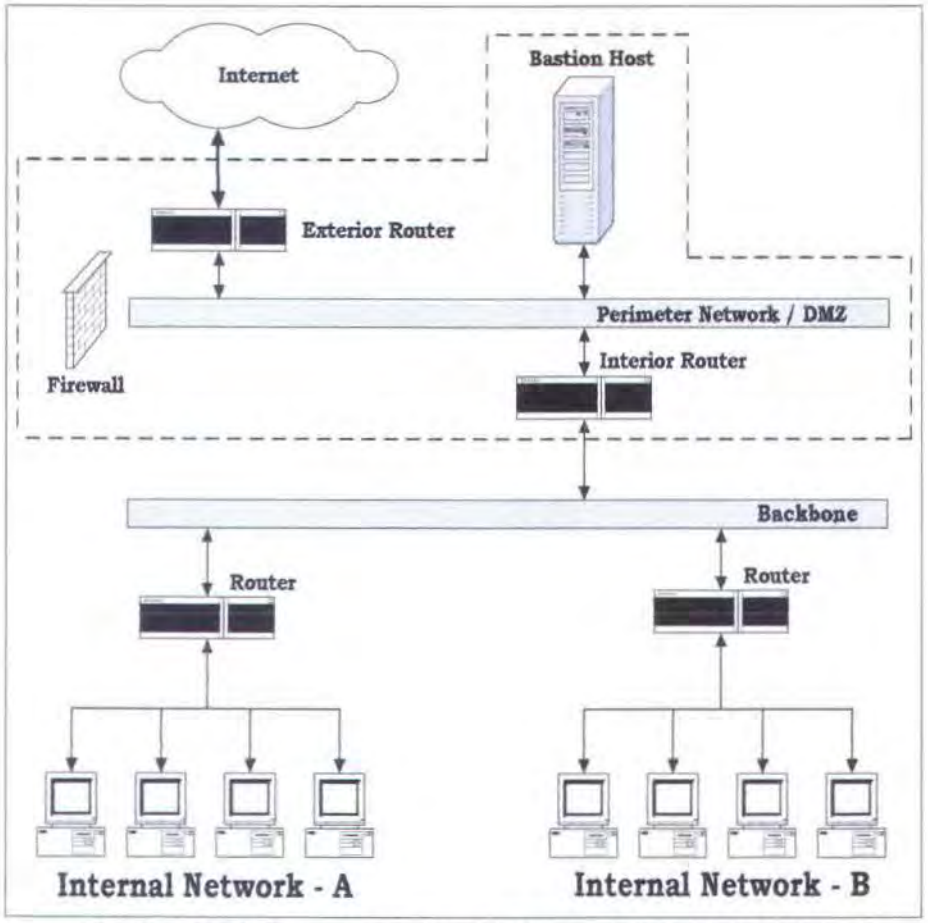
2.10.4 Penggunaan Banyak Internal Network

Berbasis pada arsitektur *screened subnet* yang memisah *internal network* dengan cara memanfaatkan jumlah *interface* pada *interior router*. Kombinasi ini sering dipakai pada suatu jaringan internal yang cukup besar dan sistem keamanan yang cukup ketat, seperti ditunjukkan pada **gambar 2-12** dan **gambar 2-13**.

Semakin banyak perangkat keras yang dipakai sebagai *firewall* sebenarnya tidak direkomendasikan karena akan mempersulit *administrator* dalam proses pemeliharaan. *Administrator* harus mengkonfigurasi tidak dalam satu waktu dan berbeda pada setiap perangkat *firewall*. Tapi keuntungannya adalah jaringan internal akan sangat terlindungi dari serangan yang berasal dari luar.



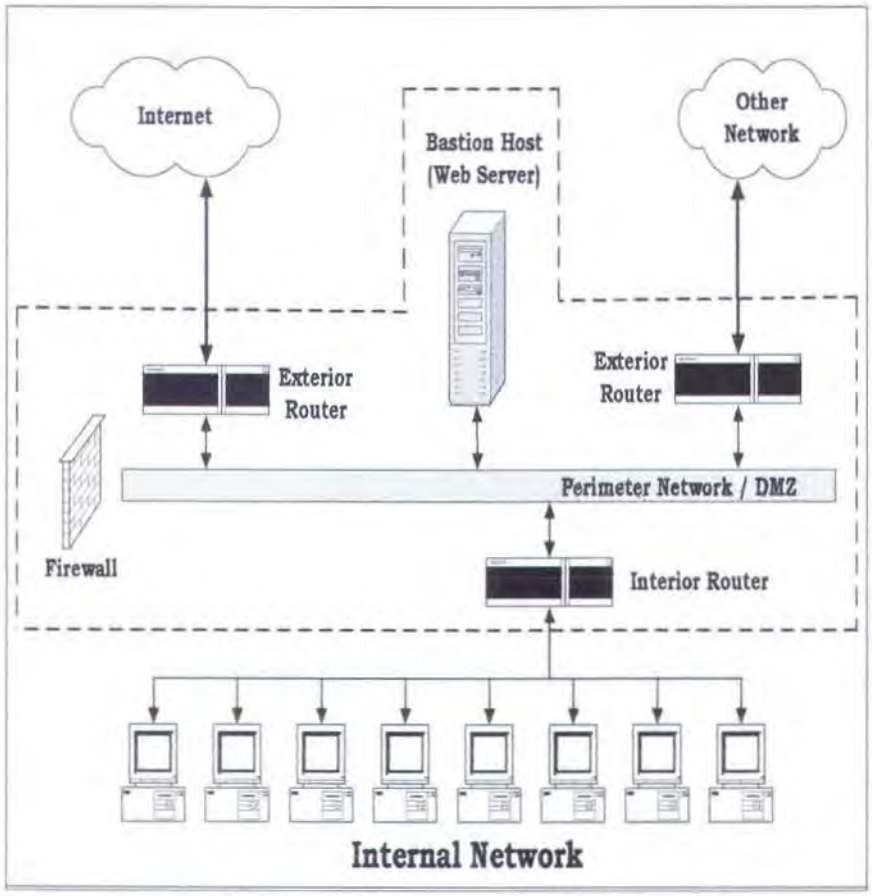
Gambar 2-12: Penggunaan banyak *Internal Network* bentuk ke-1.



Gambar 2-13: Penggunaan banyak *Internal Network* bentuk ke-2.

2.10.5 Penggunaan Banyak Exterior Router

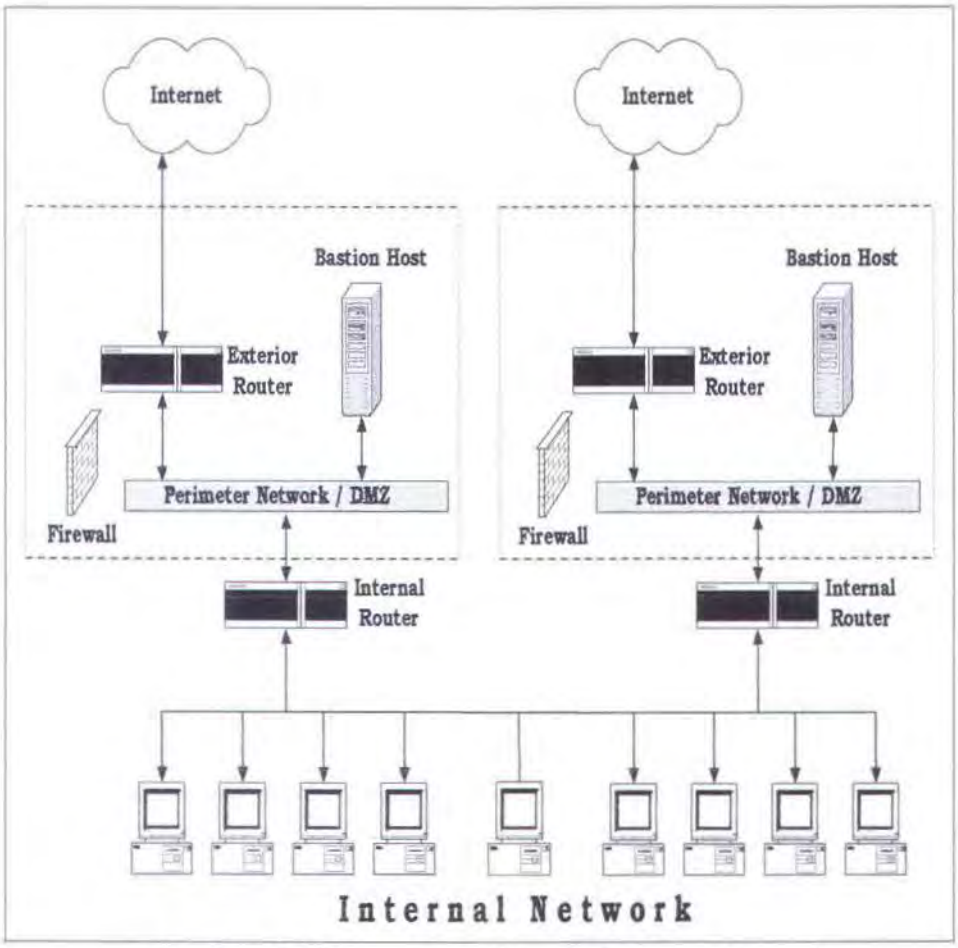
Berbasis pada arsitektur *screened subnet* dengan menambah jumlah *exterior router* untuk menghubungkan jaringan internal dengan dua atau lebih jaringan luar yang berbeda. Pada **gambar 2-14** ditunjukkan bahwa kedua *exterior router* bisa saling *redundancy* apabila ada salah satu jaringan luar terputus. Hal ini sangat efektif bagi sebuah *site* yang memerlukan koneksi tanpa terputus selama 24 jam penuh.



Gambar 2-14: Penggunaan banyak *Exterior Router.*

2.10.6 Penggunaan Banyak Perimeter Network

Berbasis pada penggunaan banyak *exterior router* yang dikembangkan dengan penambahan *perimeter network* sebagai pelengkap. Apabila hanya dengan banyak *exterior router* dirasa kurang lengkap untuk *redundancy*, maka dengan biaya yang sedikit mahal ditambah jumlah *perimeter network*. Hal ini bisa dilihat pada **gambar 2-15**.



Gambar 2-15: Penggunaan banyak *Perimeter Network*.

Selain sebagai fungsi *redundancy*, kombinasi arsitektur ini bisa sebagai pemisah antara *perimeter network* yang bersifat umum dan *perimeter network* yang bersifat *private*. Model seperti ini benar-benar aman tapi sangat boros biaya karena harus membutuhkan banyak perangkat keras.

BAB III
PERANCANGAN SISTEM
(Perangkat Firewall Dalam Sebuah Embedded PC)

3.1 Mengapa Membuat Perangkat Firewall

Banyak orang dan perusahaan yang sudah terbiasa dengan penggunaan internet atau minimal sudah mengenalnya. Mereka memerlukan suatu cara untuk membatasi akses antara internet dan komputer pribadinya maupun dari internet ke wilayah jaringan dari suatu perusahaan. Mereka membutuhkan *firewall* untuk bisa mewujudkan hal itu. *Firewall* akan melindungi komputer pribadi atau wilayah jaringan mereka dari bahaya yang muncul dari internet.

Kekhawatiran yang muncul akibat semakin bahayanya serangan orang-orang tidak bertanggung-jawab lewat media internet mendorong penulis untuk segera mewujudkan perangkat *firewall* ini. Serangan-serangan yang muncul semakin hari semakin berbahaya, semakin serius, dan semakin kompleks. Perlindungan terhadap sistem mereka, seperti disebutkan pada alinea pertama, akan lebih praktis apabila ada suatu perangkat yang berfungsi sebagai *firewall.*

Perangkat *firewall* ini harus sangat efektif dalam melindungi sebuah *site* dari kemungkinan serangan orang tidak bertanggung-jawab. Oleh karena itu penggunaan *firewall* sebagai bagian dari sistem keamanan sangat direkomendasikan. Alat ini menjadi sangat vital karena dalam satu perangkat yang terpusat mampu melindungi suatu wilayah jaringan yang terdiri dari satu atau banyak komputer sekaligus.

3.2 Pemakai Firewall

Beberapa pihak yang sangat membutuhkan *firewall* adalah sbb:

- *System/Network Administrator.* Karena pemakainya adalah orang yang sudah sangat mengenal seluk beluk jaringan, maka perangkat *firewall* ini harus mampu membuat mereka percaya dan yakin bahwa *site*-nya telah aman.
- Pengguna pribadi atau perusahaan yang telah terhubung dengan internet. Mereka diasumsikan telah mengenal sedikit mengenai apa itu jaringan komputer. Perangkat *firewall* yang mereka butuhkan adalah perangkat yang cukup mudah untuk dikonfigurasi dan mampu menjamin bahwa *site*-nya akan aman dengan adanya perangkat *firewall* ini.
- Pengguna pribadi atau perusahaan yang mulai tertarik dengan internet. Mereka umumnya belum begitu tahu mengenai manfaat dari *firewall* ini. Oleh karena itu perlu ada suatu *user education* kepada mereka untuk memandu membuat suatu sistem keamanan jaringan dengan *firewall.*
- Pengguna pribadi atau perusahaan yang sama sekali belum berpengalaman dengan internet. Hal yang paling mudah adalah memberikan panduan yang bersifat *instant* dan sangat praktis agar mereka dengan mudah bisa menggunakan *firewall* pada sistem keamanan jaringannya.

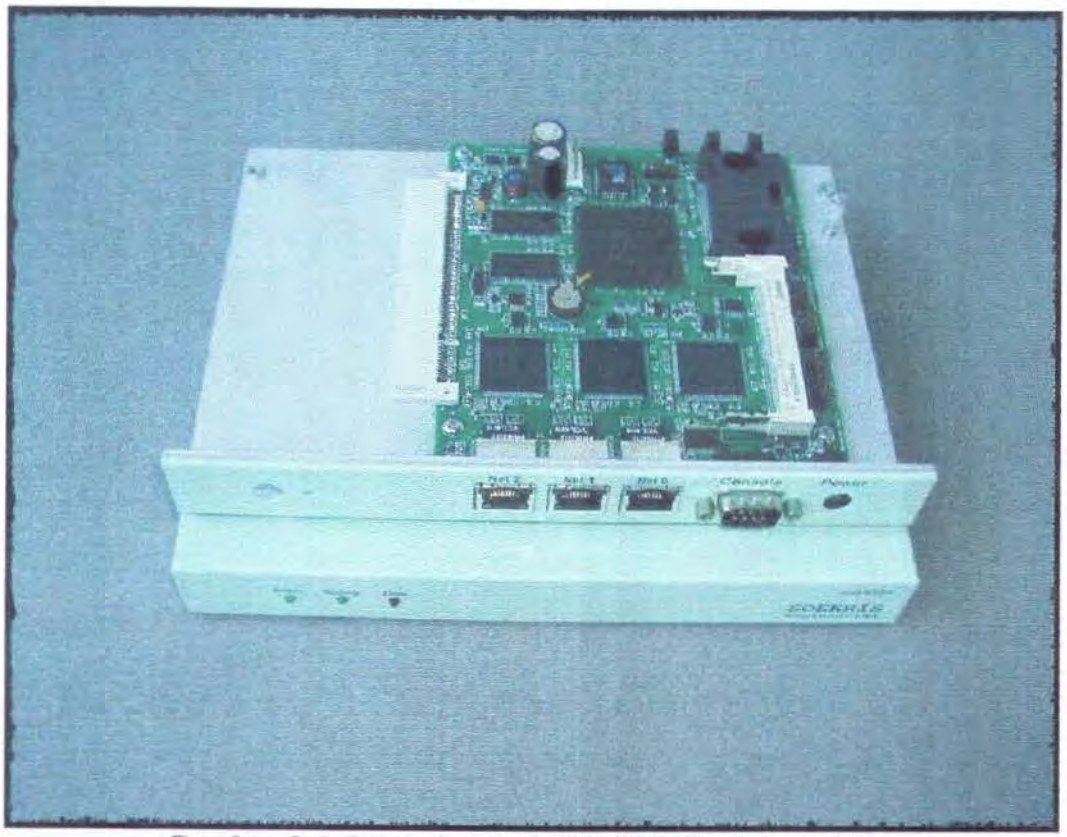
3.3 Pemilihan Perangkat Keras

Ada beberapa pilihan untuk perangkat keras yang akan dipakai. Salah satu yang sangat umum dan mudah dicari adalah PC. Dengan menggunakan PC, maka pembuatan *firewall* ini akan lebih mudah dan praktis. Dengan spesifikasi PC

bekas dengan arsitektur minimal *i386* dan asalkan bisa dipasang *ethernet card*, maka pembuatan *firewall* sudah bisa dilakukan. Tapi ada satu masalah yang cukup pelik disini, yaitu distribusi PC sebagai sebuah produk.

PC akan sangat sulit didistribusikan karena selain bentuknya yang cukup besar dan berat, juga karena sulitnya *maintenance* yang dilakukan oleh vendor. Disini penulis bertindak sebagai vendor, maka akan sulit memberikan jaminan bahwa perangkat keras, dalam hal ini adalah PC biasa, tidak akan mengalami kerusakan karena PC yang digunakan adalah bekas. Dan apabila memakai PC yang baru, selain juga sama-sama berat dan besar, harganya juga tidak seimbang dengan kebutuhannya.

Maka perangkat keras yang sangat ideal dan mendukung pembuatan *firewall* ini adalah *Embedded* PC. Seperti terlihat pada **gambar 3-1**, PC ini sangat istimewa dibandingkan dengan PC biasa. Dengan ukuran yang sangat kecil tapi mempunyai kemampuan yang tidak kalah dengan PC biasa. *Embedded* PC ini dibuat oleh vendor yang ada di Amerika dan bermerk Soekris Engineering dengan seri produk net4501. Harga *Embedded* PC ini tidak terlalu mahal dan bahkan lebih murah daripada PC biasa yang baru. Spesifikasi *Embedded* PC ini sangat efektif dan efisien untuk kebutuhan perangkat *firewall* karena memang didesain untuk pembuatan *network peripheral.*



Gambar 3-1: Perangkat *Embedded* PC: Soekris net4501.

Perangkat tersebut mempunyai dimensi 21,5 x 15 x 2,8 cm. Dan untuk lebih jelasnya, spesifikasi perangkat keras dari *Embedded* PC tersebut adalah sbb:

- *100/133 Mhz AMD ElanSC520.*
- *64 Mbyte SDRAM, soldered on board.*
- *1 Mbit BIOS/BOOT Flash.*
- *16 Mbtye CompactFLASH.*
- *3 Ethernet 10/100 Mbit ports (RJ-45 Connector).*
- *1 Serial port (DB9 pin).*
- *3 LEDs: Power, Activity, Error.*
- *Mini-PCI type III socket.*

- *PCI Slot, right angle 3.3V only.*
- *Hardware watchdog.*
- *Board size 4.85" x 5.7".*
- *Power either 5V DC fixed or 7-20V DC, max 10 Watt.*
- *Operating temperature 0-60 °C.*

Perangkat ini juga disertai dengan perangkat lunak yang sudah *add-one* di dalam BIOS-nya, yaitu:

- *comBIOS for full headless operation over serial port.*
- *PXE boot rom for diskless booting.*

3.4 Nilai Tambah pada Embedded PC

Perangkat *Embedded* PC hanyalah sebuah benda yang tidak bisa berarti apa-apa jika tidak ada perangkat lunak yang ditambahkan. Dalam hal ini FreeBSD dipilih sebagai sistem operasi dasar. Dengan sistem operasi ini secara otomatis akan didapatkan kernel yang sudah mendukung fungsi-fungsi jaringan dan variannya. Dan juga sistem operasi FreeBSD sudah mempunyai fungsi *teletype* yang bisa dimanfaatkan oleh perangkat *Embedded* PC tersebut agar bisa diakses lewat kabel *serial* untuk mendapatkan terminal atau *console*.

Sebagaimana sebuah sistem operasi yang ter-*install* pada sebuah PC biasa maka sistem operasi FreeBSD yang ter-*install* pada *Embedded* PC juga bisa ditambahi dengan aplikasi-aplikasi pendukung yang akan membantu pengembangan alat ini sebagai sebuah produk *firewall*.

Perangkat *Embedded* PC harus bisa diakses, selain dari *serial console*, dari *network interface* agar pemakaiannya, untuk mengkonfigurasi *firewall*, bisa lebih fleksibel. Untuk itu dipilih sebuah aplikasi *webserver* yang bisa mendukung hal ini. Dan selain itu, beberapa aplikasi juga harus ada untuk melengkapi *interface* ke pengguna lewat protokol TCP/IP ini. Ada beberapa aplikasi untuk mendukung pembuatan produk *firewall* ini sebagai berikut:

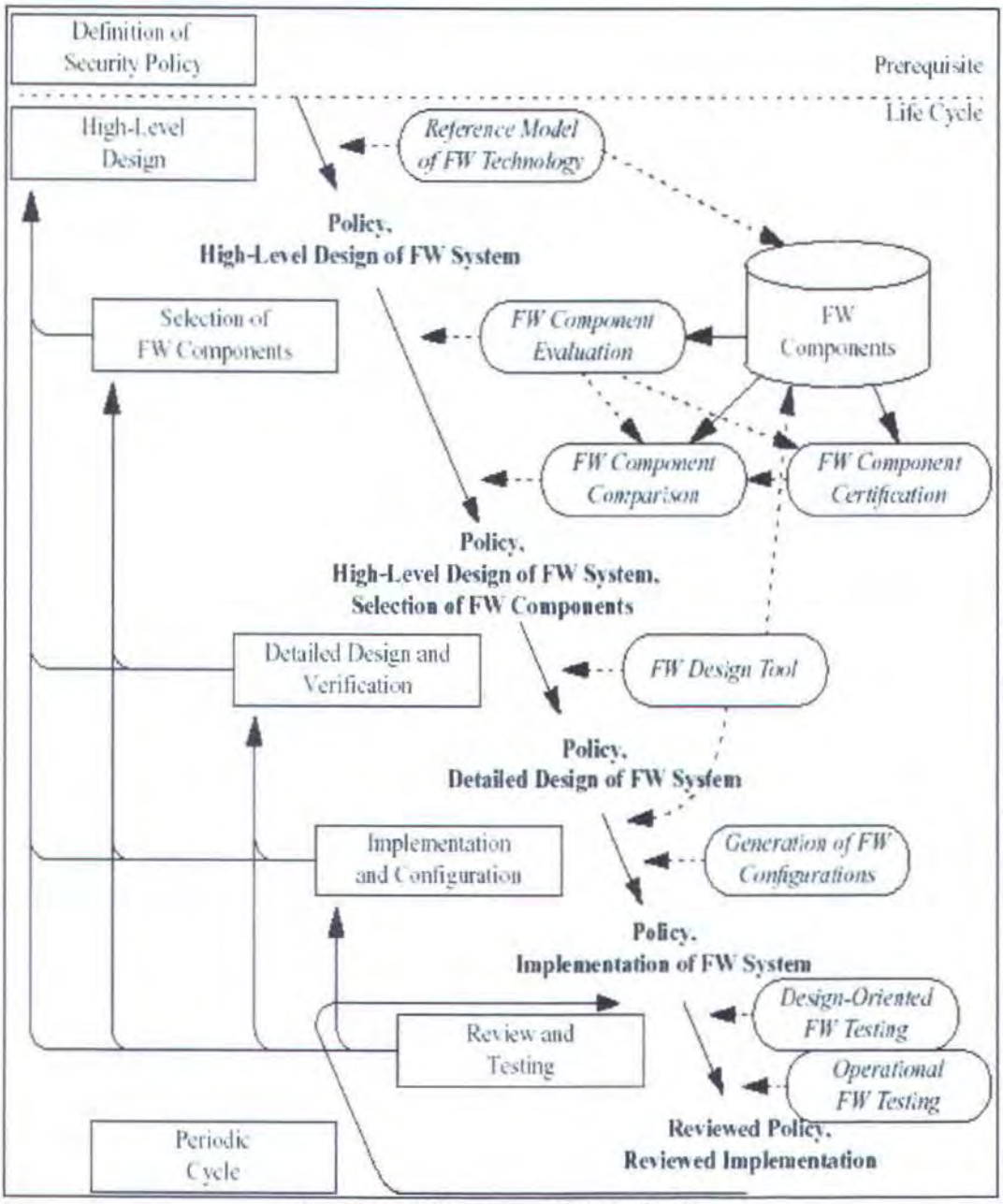
- sh shell script.
- php4 shell script.
- php4-cgi.
- thttpd webserver.
- dns_masq.

3.5 Firewall Life Cycle

Pembuatan perangkat lunak, dalam hal ini *firewall*, pasti akan mengalami perubahan dan evaluasi secara berkala serta terus-menerus. Oleh karena itu diperlukan suatu metode dalam rekayasa perangkat lunak, yaitu *software life cycle*. Disini penulis menggunakan metode *waterfall*, perlu diketahui bahwa metode ini lahir dari hasil eksperimen terhadap suatu *software* yang sangat besar bernama SAGE.

Ilustrasi pada **gambar 3-2** menunjukkan *life cycle* dari aplikasi ini, dengan pemahaman: tulisan dalam kotak di sisi sebelah kiri adalah fase, di sisi sebelah kanan dengan tulisan tercetak miring adalah metode, dan di tengah merupakan hasil (tertulis dengan huruf tebal) dari fase yang menggunakan metode disebelah

hasil (tertulis dengan huruf tebal) dari fase yang menggunakan metode disebelah kanannya. *Waterfall* sebagai suatu metode dari rekayasa perangkat lunak bersifat *continue*. Jadi, perputaran ini berlangsung terus-menerus dan tidak pernah berhenti. Setiap perputaran secara periodik akan terus me-*review* sistem dan terkadang dilakukan evaluasi serta modifikasi apabila memang diperlukan.



Gambar 3-2: *Firewall Life Cycle.*

Untuk lebih memperjelas lagi mengenai *firewall life cycle*, maka akan dijelaskan maksud dari masing-masing fase pada **gambar 3-2** sbb:

- *Phases* atau fase, adalah langkah yang secara umum menggambarkan proses pembuatan produk *firewall*.
- *Definition of Security Policy*, adalah suatu langkah awal yang merupakan syarat dari proses secara keseluruhan. Dengan langkah ini maka akan didapat: apa saja yang akan diproteksi, bagian mana yang diperbolehkan untuk melakukan akses ke atau dari internet, bagaimana prosedur operasionalnya, dan siapa saja yang bertanggung-jawab terhadap sistem ini.
- *High-Level Design*. Maksud dari *high-level design* adalah melakukan spesifikasi segala macam kebutuhan dengan agak detail. Hasilnya adalah arsitektur aplikasi *firewall* yang akan dibuat.
- *Selection of Firewall Components*. Segera setelah *high-level design* terdokumentasikan, maka bisa dipilih beberapa variasi diantara komponen-komponen firewall yang ada. Pada fase ini diharapkan mendapatkan hasil pemilihan komponen yang sesuai karena akan dipakai pada fase selanjutnya.
- *Detailed Design and Verification*. Dilakukan setelah komponen yang akan dipakai dalam sistem *firewall* sudah konkret dan sesuai dengan *high-level design*. Selain itu prosedur operasional dan konfigurasi jaringan juga harus didefinisikan dengan baik. Karena pada fase ini harus menghasilkan desain dengan lebih detail.
- *Implementation and Configuration*. Selama pada waktu dibuat dan dikonfigurasi, maka fase ini menghasilkan prosedur operasional *firewall*.

- *Review and Testing.* Segera sesudah *firewall* selesai dibuat dan dipasang, maka sebelumnya harus ada *testing* untuk melakukan validasi apakah *firewall* ini mampu berfungsi seperti yang telah didefinisikan oleh *security policy.*
- *Periodic Cycle.* Dalam implementasinya pasti *firewall* akan mengalami beberapa perubahan serta dilakukan tes dan *review*. Tujuan dari fase ini untuk selalu meninjau ulang apakah sistem yang diterapkan dan *firewall* sebagai tumpuan keamanan mampu berjalan dengan baik. Dan akan dilakukan improvisasi pada implementasinya apabila terjadi kelemahan.

Khusus untuk *Firewall Components*, diilustrasikan sebagai sebuah *database* yang berisi semua komponen *firewall* yang ada. Sedangkan untuk masing-masing metode pada **gambar 3-2** adalah sbb:

- *Reference Model of Firewall Technology.* Merupakan deskripsi fungsional dari sebuah studi terhadap macam aplikasi yang akan atau telah digunakan sehari-hari. Dengan begitu akan memberikan pemahaman tentang:
  - Fungsi apa saja yang perlu disediakan oleh *firewall.*
  - Bagaimana fungsi-fungsi itu bisa digunakan.
  - Apa saja pengaruh yang bisa ditimbulkan.
  - Apakah hasilnya seimbang dengan usaha yang telah dikeluarkan.
  - Apakah bisa diterapkan pada semua arsitektur *firewall.*
- *Firewall Component Evaluation.* Metode ini dipergunakan untuk mengevaluasi komponen-komponen *firewall* yang dipakai. Tujuan akhirnya adalah untuk mendapatkan *Firewall Component Certification* dan *Firewall*

Component Comparison. Beberapa kategori dasar yang dipakai pada metode ini adalah:

- Identifikasi produk.
- *User education* dan dokumentasi.
- Fungsionalitas.
- Opersional dan pemeliharaan atau *maintenance.*
- Biaya.

Setiap kategori diatas mengandung banyak pertanyaan, kriteria, eksperimen, dan investigasi terhadap produk *firewall* yang lain. Secara sistematis pengujian terhadap produk *firewall* adalah dengan kriteria diatas, satu dari kriteria tersebut akan memberikan pengertian yang cukup luas terhadap kualitas dari produk ini. Satu pertanyaan mungkin bisa dijawab oleh seorang konsultan bisnis, tapi pertanyaan lain baru bisa terjawab dengan riset dan pengetahuan di banyak hal.

- *Firewall Component Certification.* Ide dibalik sertifikasi ini adalah untuk mengevaluasi produk *firewall* beserta sistemnya dengan menggunakan prosedur *testing* yang sudah ditetapkan oleh lembaga yang berkepentingan. Salah satu lembaga yang bisa memberikan sertifikasi adalah *National Computer Security Association* (NCSA) di Amerika. Produk *firewall* akan menerima sertifikasi apabila telah dinyatakan lolos tes untuk memberikan jaminan kelayakan kepada calon pengguna.
- *Firewall Component Comparison.* Adalah studi komparatif terhadap komponen-komponen *firewall.* Studi ini cukup sulit karena menggunakan

matriks dua dimensi dengan kolom adalah hasil evaluasi dari kriteria dan sebagai barisnya adalah komponen-komponen *firewall.* Ada suatu hasil studi mengenai hal ini yang secara periodik di-*update* dan dipublikasikan oleh *Computer Security Institute* (CSI).

- *Firewall Design Tool.* Metode ini sebagai pendekatan untuk mengeksplorasi mekanisme *firewall* secara fungsional. Disini akan membedah sistem *firewall* secara kompleks dan meliputi semua komponen yang ada di dalamnya.
- *Generation of Firewall Configurations.* Pada metode ini akan dicobakan beberapa konfigurasi yang bisa diterapkan pada kebutuhan yang sama. Dan dari beberapa konfigurasi yang telah dicoba, maka akan dicari dan ditetapkan sebagai yang terbaik.
- *Design Oriented Firewall Testing.* Metode merupakan *testing* secara manual maupun otomatis dari konfigurasi *firewall* dan kualitas yang ditunjukkan oleh *firewall* komponen pada sebuah jaringan terintegrasi. Metode ini memiliki langkah-langkah sbb:
  - Membandingkan implementasi apakah sudah sesuai dengan rencana awal.
  - Pengecekan langsung pada *user interface* dengan hasil konfigurasi.
  - Testing pada operasional produk apakah filternya sudah cukup bagus.
  - Pengecekan terhadap beberapa aplikasi tambahan apakah sudah memerlukan *patch* atau belum.
  - Me-*review* produk secara periodik yang akan menghasilkan:
    - *Review* terhadap sistem keamanannya.
    - *Review* terhadap implementasinya.

- Beban yang ditangani oleh *firewall*.

- *Operational Firewall Testing.* Metode ini banyak dikenal dengan nama lain: *penetration testing, firewall testing*, atau bahkan *tiger team approach.* Ide di balik metode ini adalah mencari sifat kelemahan pada produk *firewall.* Cara ini bisa dilakukan secara berulang-ulang untuk melakukan validasi kelayakan. Dengan begitu akan terus memberikan koreksi untuk *administrator*, minimal memastikan keadaan *firewall*, dalam menjalankan tugasnya.

3.6 Pemilihan Komponen Firewall

Pemilihan ini didasarkan pada kebutuhan penggunaan saat ini yang sudah semakin kompleks. Beberapa hal yang sangat dibutuhkan dan bersifat krusial pada penggunaan *firewall* adalah:

- *Route the Packet.* Produk *firewall* ini harus mempunyai kemampuan seperti *router*: mem-*forward* setiap paket data yang lewat.
- *Packet Filtering.* Komponen ini mempunyai kemampuan untuk menyeleksi paket data yang lewat berdasarkan IP *source* atau *destination* dan juga berdasarkan *port number.*

- *Network Address Translation.* Kemampuan ini adalah untuk membungkus satu atau banyak IP milik jaringan internal dan menggantikannya dengan IP milik *firewall.*
- DNS *Forwarder.* Fungsi dari komponen ini mirip dengan *proxy* yang akan meneruskan *request* dari jaringan internal ke sebuah DNS *Server.*

Semua komponen ini sebagian besar sudah didukung oleh sistem operasi FreeBSD, hanya komponen untuk DNS *Forwarder* saja yang harus diambil dari luar sistem operasi ini.

3.7 Skenario Perangkat Firewall

Perangkat ini adalah sebuah produk *firewall.* Perangkat ini akan diberi nama layaknya sebuah produk komersial, mampu mendukung kebutuhan keamanan komputer dengan *software* yang *up to date*, dan tentu saja harus mempunyai sistem navigasi yang memudahkan pengguna.

3.7.1 Nama Produk Firewall

Produk *firewall* ini akan diberi nama **"GENI Firewall"**.

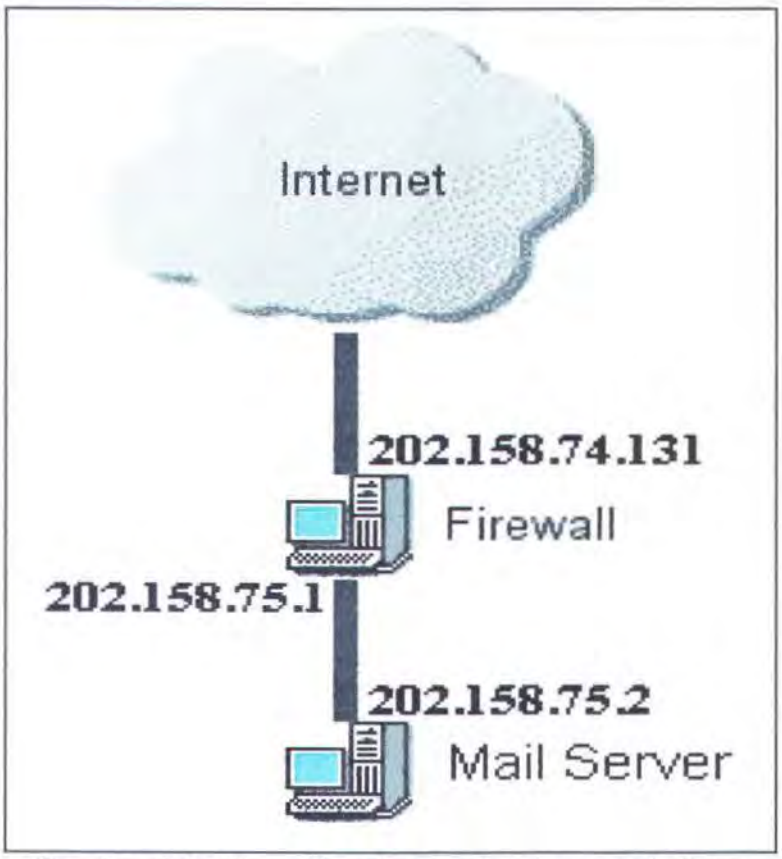
3.7.2 Desain dan Arsitektur Firewall

Produk *firewall* ini menggunakan desain *packet filtering* dan berarsitektur *screened host* yang bisa divariasi dengan menggabungkan *exterior* dan *interior router.* Desain dan arsitektur ini dipilih karena sangat mudah untuk diimplementasikan pada sebuah jaringan.

3.7.3 Dukungan Packet Filter pada FreeBSD

Pada bahasan ini akan sedikit diulas mengenai cara penggunaan fasilitas *ipfiltering* dalam sistem operasi FreeBSD. Secara *default*, *kernel* sistem operasi ini sudah mempunyai fasilitas untuk *ipfiltering* yang bisa dikonfigurasi oleh *system administrator* dengan menggunakan perintah `ipf`. Perintah `ipf` ini akan langsung mempunyai akses ke dalam *kernel* yang memang sudah mengandung fungsi-fungsi *network* yang cukup kompleks.

Seperti ditunjukkan pada **gambar 3-3** bahwa ada sebuah jaringan sederhana yang menghubungkan sebuah *mail server* ke internet. Jaringan tersebut dilindungi oleh sebuah *firewall* dengan desain *packet filtering* dan berarsitektur *dual-homed host* serta berfungsi sebagai *screening router*.



Gambar 3-3: *Firewall* pada sebuah jaringan sederhana.

Ada dua *network interfaces* yang terpasang pada *firewall*, yaitu `sis0` (terhubung ke internet) dan `sis1` (terhubung ke *mail server*).

Hal pertama yang dilakukan oleh *firewall* adalah men-*screening* paket data yang lewat berdasarkan alamat IP dan *port address*. Sebagai contoh misalnya *firewall* akan menolak semua paket data yang lewat dari internet menuju ke *mail server*, maka perintah yang harus diketik adalah:

```
ipf block in quick on sis0 from any to 202.158.75.2/32
```

Dan sebagai contoh kedua misalnya *firewall* hanya memperbolehkan akses dari alamat 202.43.171.0/24 menuju ke *mail server*. Maka perintah yang harus diketik adalah:

```
ipf pass in quick on sis0 from 202.43.171.0/24 to 202.158.75.2/32
```

Setelah dua contoh diatas yang menunjukkan tentang *screening* berdasarkan alamat IP, sekarang *firewall* akan melakukan *screening* berdasarkan alamat IP beserta *port address*. Misalnya, *firewall* hanya akan memperbolehkan akses dari internet menuju ke *mail server* pada *port* 25 saja. Maka perintah yang harus diketik adalah:

```
ipf pass in quick on sis0 proto tcp from any to 202.158.75.2/32 port = 25
```

Di dalam fitur *ipfilter* milik FreeBSD ini juga bisa untuk melakukan *network address translation*. Sebagai contoh misalnya semua paket data yang keluar dari *mail server* menuju ke internet alamatnya akan diganti dengan alamat IP milik *firewall* dengan perintah:

```
ipf map sis0 202.158.75.2/32 -> 202.158.74.131/32
```

atau

```
ipf map sis0 202.158.75.2/32 -> 0/32
```

Perintah kedua tersebut dipakai apabila misalnya koneksi *firewall* ke internet melalui *dialup* yang alamat IP-nya selalu dinamis.

Fitur lain yang akan dimanfaatkan untuk pembuatan *firewall* ini adalah fitur untuk melihat *log* atau catatan paket data. Misalnya *firewall* akan menolak semua paket data yang lewat dari *mail server* menuju ke internet dan akan dicatat *rule*-nya, maka perintah yang harus diketik adalah:

```
ipf block in log quick on sis0 from 202.158.75.2/32 to any
```

Kita asumsikan ada suatu alamat IP dari internet, yaitu 202.43.171.9/32, yang akan mencoba untuk melakukan akses ke *mail server* pada *port* 25, maka fitur *log* ini bisa ditampilkan dengan perintah berikut:

```
# ipmon -o I
21:12:03.710018 sis0 @0:1 B 202.158.75.2,25 -> 202.43.171.9,4923
PR tcp len 20 1488 -A
```

Hasil tersebut bisa diartikan kurang lebih sbb:

- `21:12:03.710018` : Menunjukkan *timestamp* pada saat terjadi koneksi.
- `sis0` : Menunjukkan *interface* di *firewall* pada saat terjadi koneksi.
- `@0:1` : Menunjukkan nomor *rule*, yaitu pada *group* 0 dan *rule* nomor 1.
- `B` : Menunjukkan status koneksi, yaitu *blocked*.
- `202.43.171.9,4923` : Menunjukkan *destination address* dan *destination port*.
- `->` : Menunjukkan arah dari *source address* ke *destination address*.
- `202.158.75.2,25` : Menunjukkan *source address* dan *source port*.

- `PR tcp` : Menunjukkan *protocol* data tersebut adalah TCP.
- `len 20 1488` : Menunjukkan ukuran paket data.
- `-A` : Menunjukkan paket data ini adalah ACK.

Untuk lebih jelas mengenai *ipfilter* pada sistem operasi FreeBSD ini, silakan melihat dokumentasinya di *situs* http://www.obfuscation.org/ipf/ yang menjelaskan secara lebih mendalam dan mendetail.

3.7.4 Pembuatan User Interface

Pembuatan *user interface* untuk produk ini menggunakan program berbasis *console* dan *web* agar nanti bisa diakses dengan mudah oleh pengguna. Untuk *user interface* pada *console* dibuat dengan menggunakan *php script* untuk menampilkan menunya. Pemilihan *php4 script* ini untuk memudahkan dalam pemrogramannya karena pada *user interface* yang berbasis *web* juga dibuat menggunakan *php4-cgi*. Pada dasarnya *user interface* ini akan menghubungkan antara pengguna dengan sistem secara tidak langsung.

Mengkonfigurasi sistem ataupun *firewall rules* sangat tidak mudah apabila dilakukan dengan cara *default*, yaitu dengan *command line interface* yang banyak digunakan pada sistem operasi Unix termasuk FreeBSD. Oleh karena itu dibuatkan penghubung antar pengguna dan sistem dengan menggunakan *user interface*: *console* dan *graphical user interface.*

Untuk menampilkan *console* secara *default* sudah didukung oleh sistem operasi FreeBSD dengan fasilitas *getty*, suatu aplikasi untuk menampilkan *console* pada layar monitor. Sedangkan untuk menampilkan *graphical user interface* dibutuhkan suatu *webserver* yang sekaligus bertindak sebagai *interpreter* untuk

program *php4-cgi*. Maka dipilihlah program *thttpd webserver* untuk mendukung aplikasi ini.

3.7.5 Menu Navigasi

Prosedur navigasi produk *firewall* ini menggunakan dua cara, pertama yaitu dengan *serial console* dan kedua dengan *graphical user interface*. Masing-masing navigasi mempunyai fungsi yang berbeda. Seperti halnya produk komersial yang lain, navigasi seperti ini sudah umum dan banyak digunakan.

Pada navigasi pertama, yaitu *serial console*, pengguna akan mendapatkan lima menu dengan masing-masing adalah:

- **`Setting Ethernet`**

  Pada menu ini pengguna akan memilih *ethernet* yang digunakan dari 3 *ethernet* yang ada. Masing-masing bisa untuk *interface* LAN, WAN, dan DMZ.

- **`Setting IP Address (LAN)`**

  Pada menu ini pengguna akan menentukan alamat IP pada *interface* LAN yang nantinya akan dipakai *browser* untuk mengakses *graphical user interface*-nya.

- **`Reset Password`**

  Pada menu ini pengguna bisa me-*reset password* untuk *administrator* sistem dengan *default password* "gfw" apabila pengguna melupakan *password* yang sudah dipakai sebelumnya.

- **Load Factory Defaults**

  Menu ini disediakan untuk *emergency action* apabila terjadi kesalahan *setting* yang mengakibatkan sistem tidak diakses lewat *graphical user interface*, maka sistem akan di-*reset* ke konfigurasi *default*. Konfigurasi *default* pada sistem ini adalah:

  - Setting Ethernet: *unconfigured.*
  - Setting IP Address: *unconfigured.*
  - Administrator Password: "gfw".

- **Reboot**

  Pengguna bisa melakukan *reboot* pada sistem dengan menu ini.

Pada navigasi kedua, yaitu *graphical user interface*, pengguna akan mendapatkan tigabelas menu yang terbagi dalam lima kategori:

- **System**
  - **General Setup**

    Pada menu ini pengguna bisa mengisi *hostname*, *domain*, DNS *Server*, dan *password* dari sistem.

  - **System Status**

    Menu ini berisi informasi tentang versi *software*, tipe *hardware*, *uptime*, dan *load average*.

  - **Ethernet Status**

    Menu ini juga berisi informasi tentang status *ethernet* yang meliputi *mac address*, alamat IP, *netmask*, *gateway*, dan *packets rate*.

- **Backup / Restore**

  Pada menu ini pengguna melakukan *backup* terhadap konfigurasi sistem dengan men-*download file* konfigurasi tersebut dalam format XML. Dan begitu juga sebaliknya, dapat melakukan *upload file* konfigurasi.

- **Reboot System**

  Pengguna bisa melakukan *reboot* terhadap sistem dengan menu ini.

- **IP Addresses**

  - **LAN**

    Pada menu ini pengguna bisa mengisi alamat IP dari *interface* LAN.

  - **DMZ**

    Pada menu ini pengguna bisa mengisi alamat IP dari *interface* DMZ.

  - **WAN**

    Pada menu ini pengguna bisa mengisi alamat IP dari *interface* WAN.

- **Firewall**

  - **Rules**

    Ini adalah menu inti untuk melakukan konfigurasi *firewall*. Setiap *rules* pada setiap *interface* bisa dikonfigurasi menggunakan menu ini.

  - **Refresh State**

    Dengan menu ini pengguna bisa me-*refresh* semua koneksi yang *state*-nya sudah disimpan oleh sistem.

  - **Logs**

    Menu ini berisi informasi tentang 50 catatan terakhir dari aktivitas *firewall rules* yang telah dikonfigurasi sebelumnya.

- **Services**
  - **DNS Forwarder**

    Pada menu pengguna bisa melakukan *setting* pada DNS *Forwarder* sebagai variasi *rules*.
- **Misc**
  - **Ping Test**

    Pada menu pengguna bisa melakukan *ping test* pada suatu *host* atau alamat IP tertentu.

3.8 Instalasi FreeBSD di Embedded PC

Sebagai permulaan, kita harus melakukan instalasi FreeBSD ini pada sebuah PC biasa terlebih dahulu. Versi yang digunakan pada instalasi ini adalah FreeBSD 4.8-STABLE. Tujuan dari instalasi ini adalah membuat sebuah *image* berukuran kecil yang berisi sistem operasi FreeBSD lengkap dengan aplikasi-aplikasi pendukung untuk *firewall*.

3.8.1 FreeBSD di dalam FreeBSD

Sebagai permulaan, kita harus melakukan instalasi FreeBSD ini pada sebuah PC biasa. Kemudian membuat sebuah direktori yang nantinya akan diisi *base files* dari FreeBSD yang akan kita buat. Dengan perintah:

```
# mkdir /usr/net4501
```

kemudian masukkan CD instalasi FreeBSD lalu ketik `/stand/sysinstall`, ikuti menu "Custom Installation" lalu lakukan instalasi "Install Root" ke direktori

/usr/net4501. Jadilah direktori FreeBSD baru yang telah siap untuk dipakai, lalu ketiklah:

```
# chroot /usr/net4501
```

maka secara otomatis direktori *root* akan berpindah ke /usr/net4501. Kemudian untuk mendapatkan *shell* tambahkan baris dibawah pada /usr/net4501/root/.cshrc :

```
set prompt = "Net4501 %~ %# "
```

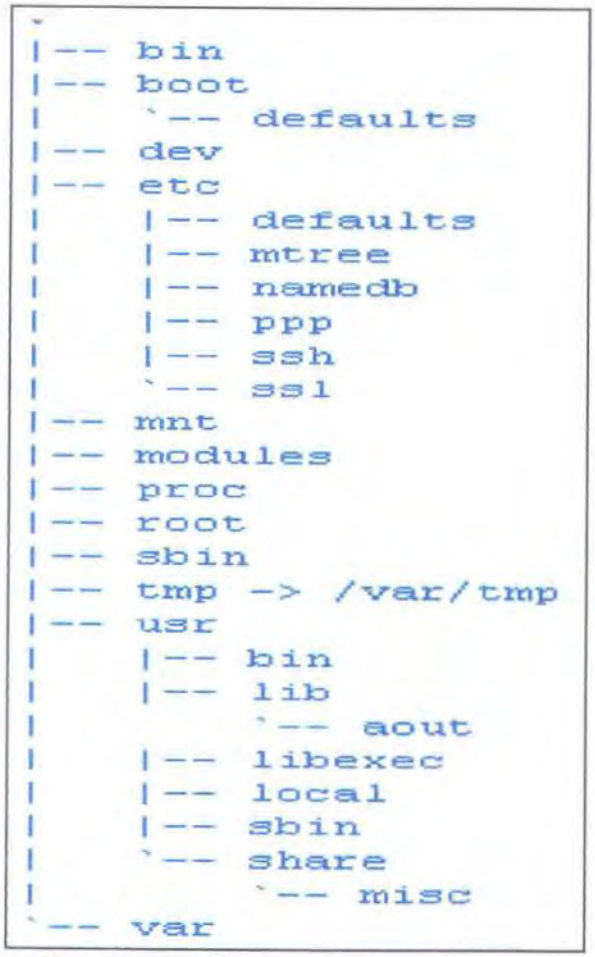
Dengan demikian maka telah kita dapatkan *shell* yang siap untuk dijalankan.

3.8.2 Menyusun Hirarki Direktori

Selanjutnya akan kita susun direktori yang hirarkinya tidak jauh berbeda dengan hirarki direktori secara *default*. Hirarki direktori ini seperti ditunjukkan pada **gambar 3-4**. Penyusunan direktori ini dimulai dengan membuat direktori baru /usr/gfw dengan perintah:

```
# mkdir /usr/gfw
```

Kemudian untuk atribut atau *permission*-nya pada setiap direktori adalah 0755 kecuali direktori /proc yang mempunyai atribut 0555.



Gambar 3-4: Susunan hirarki direktori.

3.8.3 Membuat Kernel Baru

Kernel baru harus dibuat dengan mengkompilasi ulang *source* tapi menggunakan konfigurasi yang lebih spesifik untuk *Embedded* PC ini. Pertama-tama masuk ke direktori `/sys/i386/conf` dan kemudian menyalin konfigurasi GENERIC ke GFW01 dengan perintah:

```
# cp GENERIC GFW01
```

Setelah itu tambahkan baris dibawah ini pada konfigurasi GFW01 tersebut:

```
options CLK_USE_I8254_CALIBRATION
options CPU_ELAN
```

```
options HZ=250
```

Konfigurasi diatas ditambahkan untuk menyesuaikan *kernel* dengan prosesor AMD Elan SC520 yang terdapat di dalam *Embedded* PC tersebut.

Maka kompilasi *kernel* sudah bisa dimulai dengan perintah:

```
# config -g GFW01
# cd /sys/compile/GFW01
# make depend && make
```

yang sesaat kemudian akan menjadi *kernel* baru. *Kernel* baru ini harus di-*compress* terlebih dahulu agar ukurannya menjadi lebih kecil dengan perintah:

```
# gzip -9 kernel
# cp kernel.gz /usr/gfw
```

maka selesailah tahap pembuatan *kernel* baru ini.

3.8.4 Modifikasi Isi Direktori

Agar supaya sistem ini dapat berjalan normal, maka akan ada beberapa perubahan di direktori `/etc`. Kita mulai untuk merubah konfigurasi pada `/etc/fstab` seperti dibawah ini:

```
/dev/ad0a /      ufs     ro  1  1
proc      /proc  procfs  rw  0  0
```

karena media *disk* yang digunakan adalah *Compact Flash*. Kemudian konfigurasi `/etc/rc.conf` juga harus dirubah seperti dibawah ini:

```
hostname="geni.fire.wall"
ifconfig_sis0="10.0.0.5 netmask 255.255.255.0"
kern_securelevel_enable="NO"
```

```
sendmail_enable="NONE"
sshd_enable="NO"
usbd_enable="NO"
inetd_enable="NO"
portmap_enable="NO"
update_motd="NO"
varsize="8192"
```

Setelah itu konfigurasi `/etc/ttys` juga perlu dirubah agar sistem bisa dikontrol dari *serial console.* Dari konfigurasi yang lama seperti dibawah ini:

```
ttyd0 "/usr/libexec/getty std.9600" dialup off secure
```

akan dirubah menjadi:

```
ttyd0 "/usr/libexec/getty std.9600" vt100 on secure
```

Konfigurasi `/etc/master.passwd` harus di bangun ulang setelah sebelumnya menambahkan beberapa *user* dengan perintah:

```
pwd_mkdb -p -d /usr/gfw/etc /usr/gfw/etc/master.passwd
```

Kemudian seperti kita ketahui bahwa sistem operasi Unix akan membaca suatu *device* dari konfigurasi, yang sudah didefinisikan terlebih dahulu, di direktori `/dev`. Untuk mendefinisikan konfigurasi ini bisa menggunakan *script* yang sudah disediakan oleh sistem operasi dengan perintah:

```
cp /dev/MAKEDEV /usr/gfw/dev
cd /usr/gfw/dev
sh MAKEDEV all
```

Dengan begitu secara otomatis akan terbentuk konfigurasi *device* yang nantinya akan dipakai oleh *kernel* dari sistem operasi dalam mengakses suatu *device.*

Terakhir adalah memasukkan semua aplikasi tambahan beserta program *Graphical User Interface* ke dalam direktori `/usr/gfw` agar semuanya menjadi lengkap dan siap pakai.

3.8.5 Pembungkusan Semua Direktori dan Isinya

Untuk memudahkan dalam pemindahan semua direktori dan isinya, maka harus dilakukan pembungkusan dengan perintah berikut:

```
cd /usr/gfw
tar cfvz /usr/gfw-disk.tgz *
```

Dengan demikian semua direktori dan isinya menjadi satu dalam sebuah *file* `gfw-disk.tgz`.

3.8.6 Pembuatan Image GENI Firewall

Pembuatan *image* ini adalah membuat sebuah *virtual node* di dalam *harddisk* yang nantinya akan diaplikasikan sebagai *filesystem* ke dalam *Compact Flash* milik *Embedded* PC. Jadi kita tidak akan membuat program atau menjalankan sistem operasi FreeBSD di dalam *Embedded* PC dalam keadaan belum siap pakai.

Langkah awal adalah membuat sebuah *zero device* yang semu dari *node* `/dev/zero` ke dalam sebuah *file* dengan perintah:

```
dd if=/dev/zero of=/usr/gfw-image.bin bs=512
```

Setelah terbentuk *file* `gfw-image.bin` maka *virtual node* bisa dibuat dengan perintah:

```
vnconfig -s labels -c vn0 /usr/gfw-image.bin
```

dan dengan terbentuknya *virtual node* `vn0` maka seakan-akan kita sudah mempunyai *harddisk* baru yang siap untuk dipakai.

Secara *logic* kita mempunyai *harddisk* yang nyata dan akan dipartisi dengan perintah:

```
disklabel -Brw vn0 auto
disklabel -e vn0
```

yang kemudian dijadikan *filesystem* dengan ukuran 8 *Mbyte* ke dalam *virtual node* tersebut dengan perintah:

```
newfs -b 8192 -f 1024 -U /dev/vn0a
```

Seperti halnya *device* lainnya, untuk mengakses *virtual node* tersebut harus dilakukan *mounting* ke hirarki direktori utama dengan perintah:

```
mount /dev/vn0a /mnt
```

Kemudian yang terpenting adalah memasukkan semua isi direktori `/usr/gfw` ke dalam direktori `/mnt` tersebut. Dan karena semua isi direktori `/usr/gfw` sudah dibungkus dalam *file* `/usr/gfw-disk.tgz` maka jalankan perintah berikut:

```
cd /mnt
tar xfvzP /usr/gfw-disk.tgz
```

Dan sebagai langkah terakhir adalah melakukan *unmounting* serta pelepasan *virtual node* `vn0` dari *kernel* dengan perintah berikut:

```
cd /
umount /mnt
vnconfig -u vn0
```

Maka lengkaplah sudah proses pembuatan *firewall image* ini.

3.8.7 Konfigurasi BIOS pada Embedded PC

Agar bisa berjalan seperti yang diinginkan, maka perangkat keras Soekris net4501 ini harus mempunyai konfigurasi BIOS seperti dibawah ini:

```
ConSpeed = 9600
ConLock = Enabled
BIOSentry = Disabled
PCIROMS = Disabled
PXEBoot = Disabled
FLASH = Primary
BootDelay = 2
BootPartition = Disabled
ShowPCI = Enabled
```

Untuk lebih jelas mengenai cara mengkonfigurasi BIOS pada alat ini, silakan melihat dokumentasinya di *situs* http://www.soekris.com/ yang menjelaskan secara lebih dalam dan mendetail.

3.8.8 Pemindahan Image ke dalam Embedded PC

Embedded PC tidak bisa melakukan instalasi sistem operasi pada dirinya sendiri karena tidak mempunyai media instalasi seperti PC pada umumnya. Untuk bisa mendapatkan *console* BIOS milik *Embedded* PC harus memakai kabel serial

DB9 yang dihubungkan ke PC biasa. Akses ke *Embedded* PC ini dilakukan dengan bantuan *software HyperTerninal* yang ada di dalam sistem operasi MS-Windows® dengan konfigurasi 8-n-1.

Embedded PC secara *default* sudah mempunyai PXE *Boot Environment* di dalam BIOS-nya. Dengan ini maka *Embedded* PC bisa melakukan *booting* dengan sebuah *BootServer* yang secara fisik terhubung dengan kabel UTP atau kabel *network*. Ada dua komponen penting yang harus ada di *BootServer*, yaitu `DHCP` dan `tftp`. `DHCP` adalah kependekan dari *Dynamic Host Configuration Protocol* yang berfungsi untuk memberikan sebuah alamat IP kepada komputer *client* yang memintanya. Sedangkan `tftp` adalah kependekan dari *Trivial File Transfer Protocol* yang berfungsi untuk mentransfer *kernel* dari *BootServer* ke dalam komputer *client* yang memintanya.

Untuk memudahkan instalasi ini maka *BootServer* adalah sebuah PC yang didalamnya sudah ter-*install* sistem operasi FreeBSD. Kemudian hal pertama yang harus disiapkan adalah melakukan instalasi `DHCP` di dalam *BootServer* ini dengan cara sbb:

```
# cd /usr/ports/net/isc-dhcp3
# make all install clean
```

Secara otomatis sistem operasi FreeBSD akan men-*download* aplikasi `isc-dhcp3` dari internet lalu dikompilasi dan di-*install* ke dalam sistem.

Setelah itu masukkan konfigurasi berikut ini ke dalam *file* konfigurasi isc-dhcp3 :

```
host soekris {
     hardware ethernet 00:00:24:c0:b7:88;
     fixed-address 10.0.0.100;
     filename "pxeboot";
     next-server 10.0.0.5;
     option root-path "10.0.0.5:/usr/net4501";
 }
```

Konfigurasi diatas menunjukkan bahwa *Embedded* PC mempunyai alamat fisik pada *network interface*-nya adalah 00:00:24:c0:b7:88 dan akan diberikan nomor 10.0.0.100 sebagai alamat IP-nya. Sedangkan untuk *filesystem* dan *root partition*-nya ada pada direktori `/usr/net4501`.

Kemudian yang kedua adalah menyiapkan `tftp` *server* di dalam *BootServer*. Hal ini tidak sesulit pada instalasi `dhcp`, karena hanya cukup dengan menambahkan baris berikut ini pada konfigurasi `/etc/inetd.conf` :

```
tftp  dgram  udp wait root /usr/libexec/tftpd  tftpd -s  /tftpboot
```

maka *file* `pxeboot` harus ditaruh didalam direktori `/tftpboot` dengan perintah sbb:

```
cp /usr/src/sys/boot/i386/pxeldr/pxeboot /tftpboot
```

Setelah itu me-*restart daemon* `inetd` dengan perintah berikut ini:

```
# kill -HUP `cat /var/run/inetd.pid`
```

Terakhir adalah mengkonfigurasi NFS *Server* yang dimulai dengan menambahkan baris berikut ke dalam konfigurasi `/etc/rc.conf`:

```
nfs_client_enable="YES"
nfs_reserved_port_only="YES"
nfs_server_enable="YES"
```

dan setelah itu mendefinisikan direktori yang akan dipakai untuk NFS *Server* dengan perintah berikut:

```
# cat >> /etc/exports
/usr/net4501 -ro -maproot=0 10.0.0.100
kill -HUP `cat /var/run/mountd.pid`
```

Agar *image* bisa dipindahkan dari NFS *Server* ke dalam *Embedded* PC maka *file* `/usr/gfw-disk.tgz` harus disalin dengan perintah sbb:

```
# cp /usr/gfw-image.bin /usr/net4501
```

Dengan demikian maka *BootServer* sudah siap untuk digunakan.

Embedded PC secara otomatis akan mendapatkan *shell* sistem operasi FreeBSD setelah proses *network booting* ini berhasil. Maka *firewall image* ini sudah siap untuk dipindah ke dalam *CompactFlash* yang ada di dalam *Embedded* PC. Pemindahan ini dilakukan dengan perintah berikut:

```
# dd if=/gfw-image.bin of=/dev/rad0 bs=8k
```

Maka lengkaplah sudah proses instalasi ini dan setelah di-*reboot* sistem *firewall* sudah siap dijalankan.

BAB IV
IMPLEMENTASI DAN UJI COBA

Pada bab ini akan dijelaskan mengenai implementasi *firewall*, lingkungan uji coba, hasil uji coba, serta perbandingan dengan sebuah *firewall* yang menggunakan perangkat PC biasa.

4.1 Lingkungan Uji Coba

Uji coba dilakukan di "Javatech Internet Center ITS" (JIC ITS). Perangkat *GENI Firewall* dipasang pada *rackmount* di ruang *server* "JIC ITS". Perangkat *GENI Firewall* dipasang sejajar dengan *Proxy Server* milik "JIC ITS". Perlu diketahui bahwa *Proxy Server* milik "JIC ITS" tersebut adalah *firewall* berdesain kombinasi *packet filtering* dan *proxy services* dan berasitektur *sreened subnet* dengan variasi penggabungan *exterior* dan *interior router*.

4.1.1 Perangkat Keras

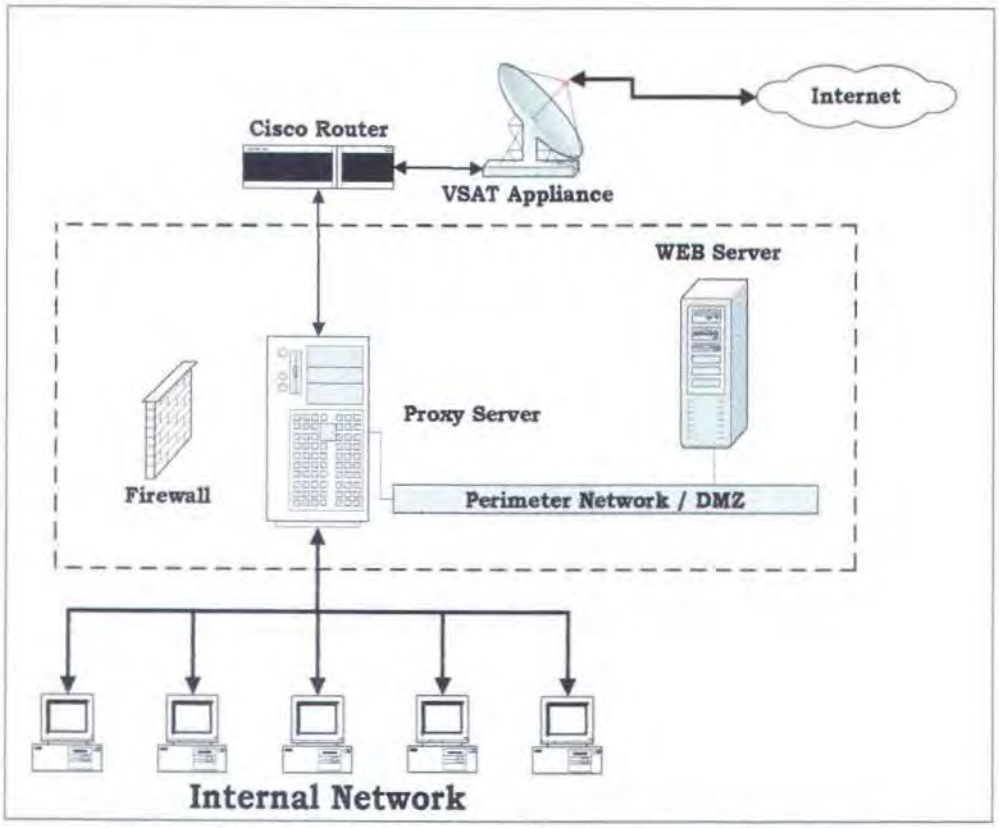
Perangkat keras yang digunakan di ruang *server* milik "JIC ITS" adalah sebagai berikut:

- *LAN Switch*, menggunakan produk *COMPEX*.
- *Proxy Server*, menggunakan spesifikasi *hardware* sbb:
  - *Intel Pentium® III 800 MHz.*
  - *IDE 20GB Harddisk.*
  - *1 Gbyte RAM.*
  - *3 Ethernet Card*, menggunakan produk *Intel EEPRO100.*

- *Web Server*, menggunakan spesifikasi *hardware* sbb:
  - *Intel Pentium® II 400 MHz.*
  - *128 Mbyte RAM.*
  - *IDE 20GB Harddisk.*
  - *Ethernet Card*, menggunakan produk *Intel EEPRO100.*
- *Router*, menggunakan produk *Cisco 2500.*

4.1.2 Topologi Jaringan

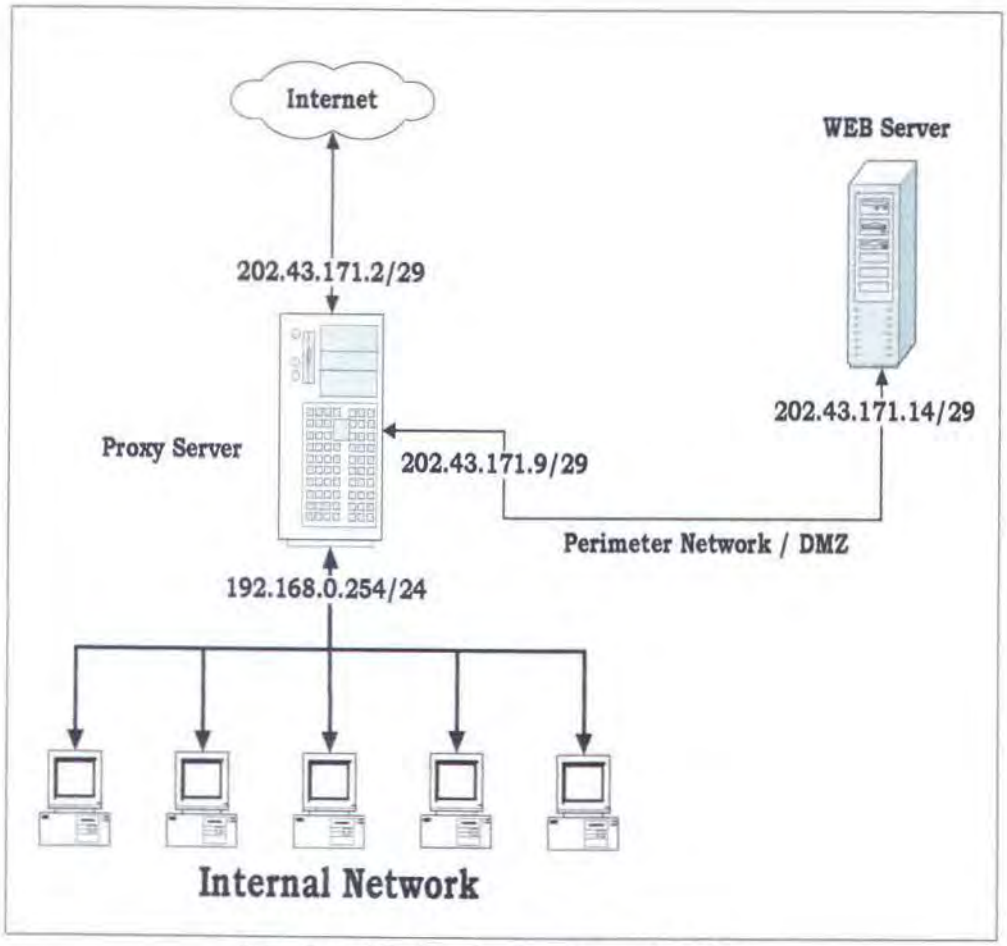
Pada **gambar 4-1** ditunjukkan topologi jaringan yang digunakan di ruang *server* milik "Javatech Internet Center ITS".



Gambar 4-1: Topologi jaringan di "Javatech Internet Center ITS".

Desain *firewall* yang digunakan adalah kombinasi *packet filtering* dan *proxy services*. Dan arsitektur *firewall* yang digunakan adalah *screened subnet* dengan variasi penggabungan *exterior router* dan *interior router*.

Sedangkan untuk alokasi IP *Address* pada masing-masing *host* akan ditunjukkan pada **gambar 4-2**. *Network Address* yang didapat oleh "JIC ITS" adalah `202.43.171.0/28`. Kemudian dibagi menjadi dua *network address* yang masing-masing adalah `202.43.171.0/29` dan `202.43.171.8/29`. Dan *network address* untuk jaringan internal adalah `192.168.0.0/24`.



Gambar 4-2: Pembagian *Network Address*.

4.1.3 Perangkat Lunak

Perangkat lunak yang digunakan pada *proxy server* milik "JIC ITS" adalah sebagai berikut:

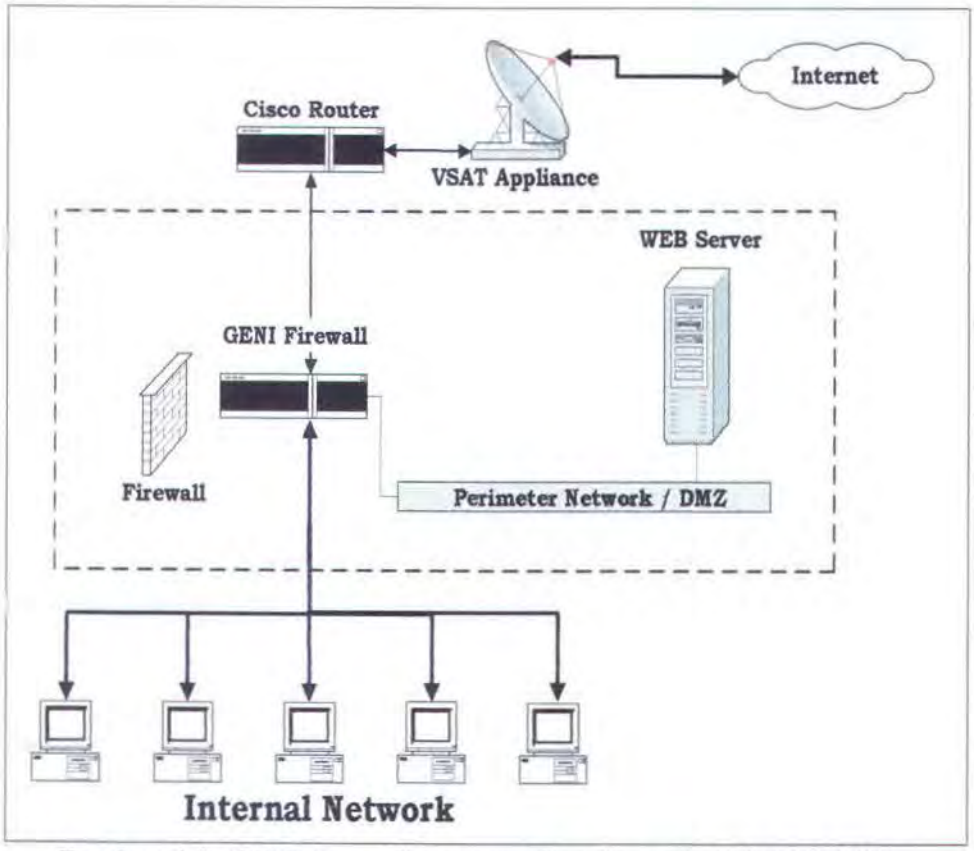
- Sistem Operasi, menggunakan FreeBSD 5-1.
- *Packet Filtering Application*, menggunakan `ipf`.
- *Proxy Server Application*, menggunakan SQUID-2.4-STABLE.
- *Remote Administrator*, menggunakan Secure-Shell (SSH).

Perangkat lunak yang digunakan pada *webserver* milik "JIC ITS" adalah sebagai berikut:

- Sistem Operasi, menggunakan Debian Linux 3-0.
- *Webserver Application*, menggunakan Apache 1-3-26.
- *Remote Administrator*, menggunakan Secure-Shell (SSH).
- *File Transfer Protocol Server*, menggunakan *ProFTPD*.
- *Mail Server*, menggunakan *Postfix*.

4.2 Instalasi GENI Firewall

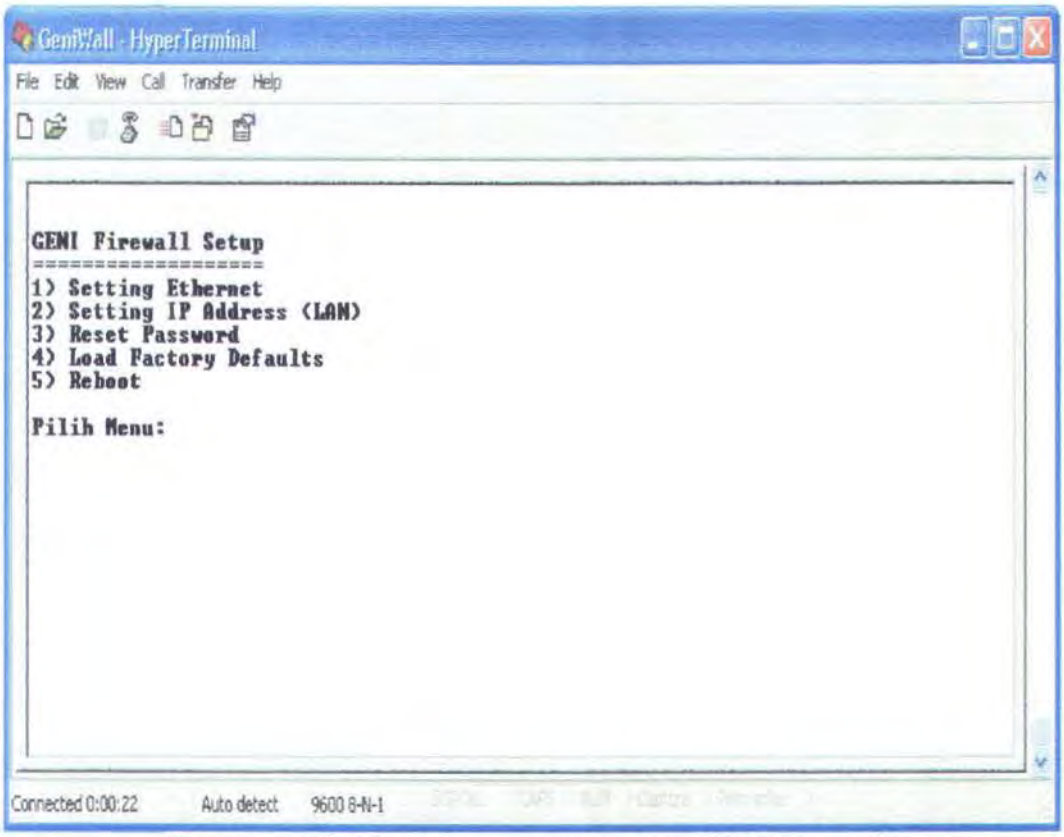
Hal pertama yang dilakukan adalah mengganti *firewall* milik "JIC ITS" dengan *GENI Firewall*, seperti terlihat pada **gambar 4-3**.



Gambar 4-3: *GENI Firewall* menggantikan *firewall* milik "JIC ITS".

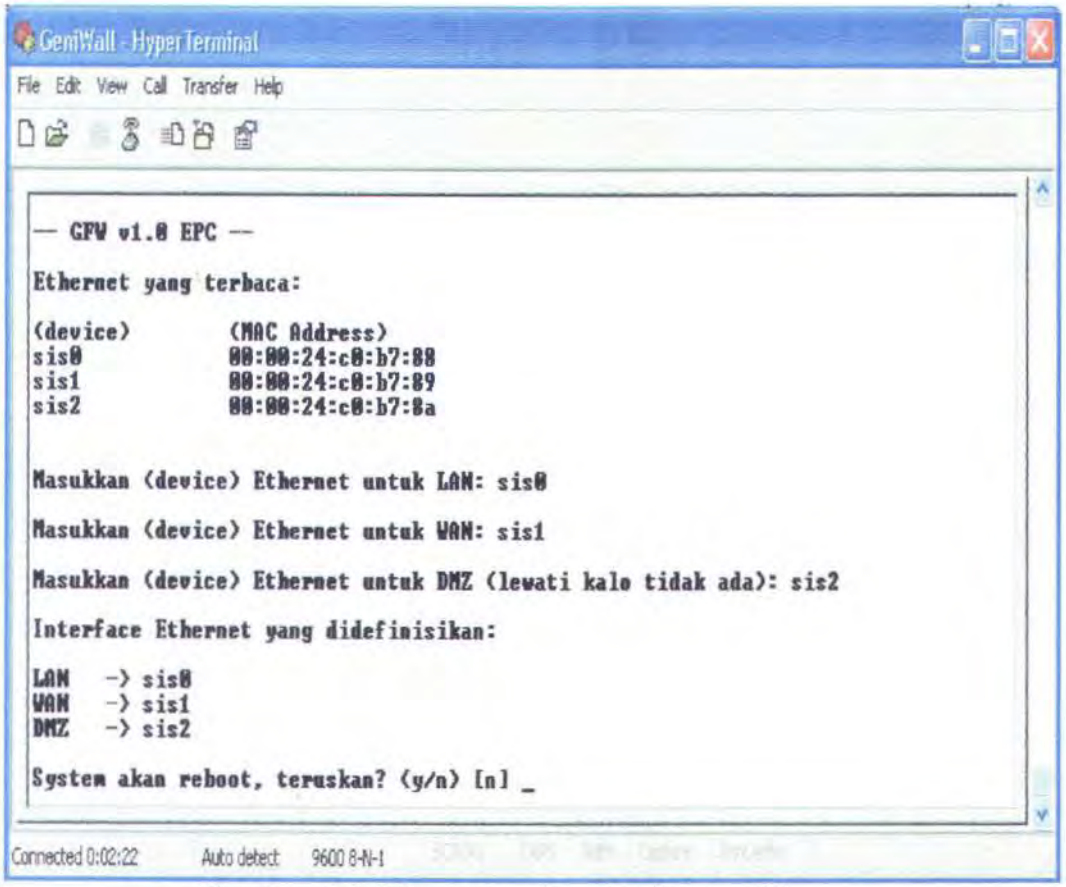
Kemudian menyamakan *setting* untuk alokasi IP *Address* dengan langkah-langkah sbb:

- Jika akses *console* pada perangkat *GENI Firewall* berhasil maka akan terlihat menu utama seperti pada **gambar 4-4**.



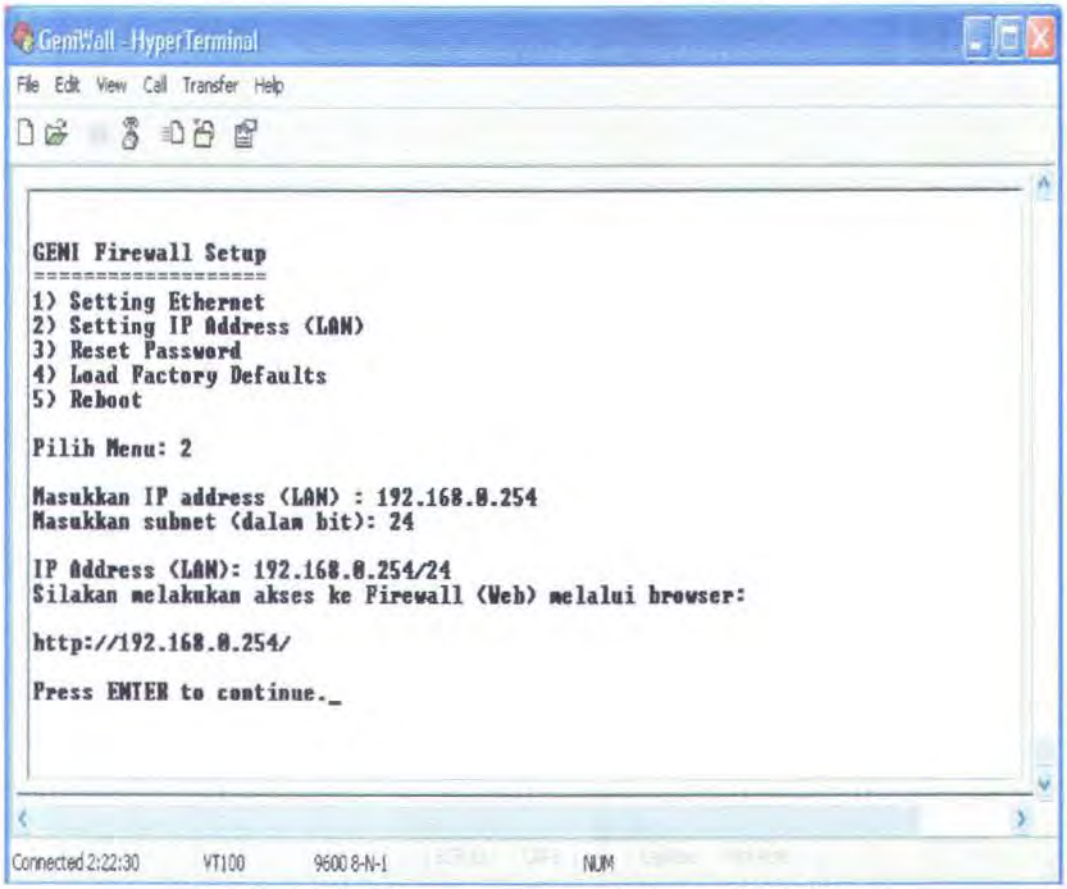
Gambar 4-4: Menu utama *console* pada *GENI Firewall.*

- Menu pertama dipilih untuk mendefinisikan *interface* yang akan dipakai seperti yang terlihat pada **gambar 4-5**.



Gambar 4-5: Menu *console* untuk *Setting Ethernet.*

- Setelah sistem melakukan *reboot*, pilih menu kedua untuk melakukan *setting* IP *Address* seperti yang terlihat pada **gambar 4-6**.



Gambar 4-6: Menu *console* untuk *Setting* IP *Address*.

Perangkat *GENI Firewall* sudah siap untuk diakses melalui *web browser* untuk mendapatkan *Graphical User Interface*. Menu ketiga sampai dengan kelima hanya digunakan apabila diperlukan saja. Jadi, agar perangkat *GENI Firewall* bisa diakses dalam *graphical user interface* cukup dengan menjalankan menu pertama dan kedua saja.

4.3 Skenario Percobaan

Skenario dari percobaan ini adalah mengimplementasikan *GENI Firewall* dengan desain *packet filtering* dan arsitektur *screend subnet* dengan variasi penggabungan *exterior* dan *interior router*. Memang ada perbedaan antara

firewall milik "JIC ITS" dengan *GENI Firewall*, yaitu bahwa *GENI Firewall* tidak bisa mengimplementasikan *firewall* dengan desain *proxy services.*

Pada **tabel 4-1**, **tabel 4-2**, dan **tabel 4-3** dibawah ini akan ditunjukkan *firewall rules* yang akan diimplementasikan pada masing-masing *interface. Rules* yang terdapat pada tabel-tabel dibawah ini menunjukkan paket data yang diijinkan untuk lewat, sedangkan yang lainnya tidak diinjinkan.

No	Protocol	Source	S-Port	Destination	D-Port	Service
1	ICMP	LAN	-	DMZ	-	ICMP
2	TCP	LAN	*	any	21	FTP
3	TCP	LAN	*	any	22	SSH
4	TCP/UDP	LAN	*	any	25	SMTP
5	UDP	LAN	*	any	53	DNS
6	TCP	LAN	*	any	80	HTTP
7	TCP/UDP	LAN	*	any	110	POP3
8	TCP/UDP	LAN	*	any	143	IMAP
9	TCP	LAN	*	any	443	HTTPS
10	TCP/UDP	LAN	*	! DMZ	5000	Y! Messenger
11	TCP/UDP	LAN	*	! DMZ	6000-7000	IRC

Tabel 4-1: Tabel *firewall rules* pada LAN *interface.*

No	Protocol	Source	S-Port	Destination	D-Port	Service
1	TCP	DMZ	20	any	*	FTP-data
2	TCP	DMZ	*	! LAN	21	FTP
3	TCP	DMZ	*	! LAN	22	SSH
4	TCP/UDP	DMZ	*	! LAN	25	SMTP
5	UDP	DMZ	*	! LAN	53	DNS
6	TCP	DMZ	*	! LAN	80	HTTP
7	TCP	DMZ	*	! LAN	443	HTTPS

Tabel 4-2: Tabel *firewall rules* pada DMZ *interface.*

No	Protocol	Source	S-Port	Destination	D-Port	Service
1	TCP	any	20	any	*	FTP-data
2	TCP	any	*	DMZ	21	FTP
3	TCP	any	*	DMZ	22	SSH
4	TCP/UDP	any	*	DMZ	25	SMTP
5	UDP	any	*	DMZ	53	DNS
6	TCP	any	*	DMZ	80	HTTP
7	TCP	any	*	DMZ	443	HTTPS

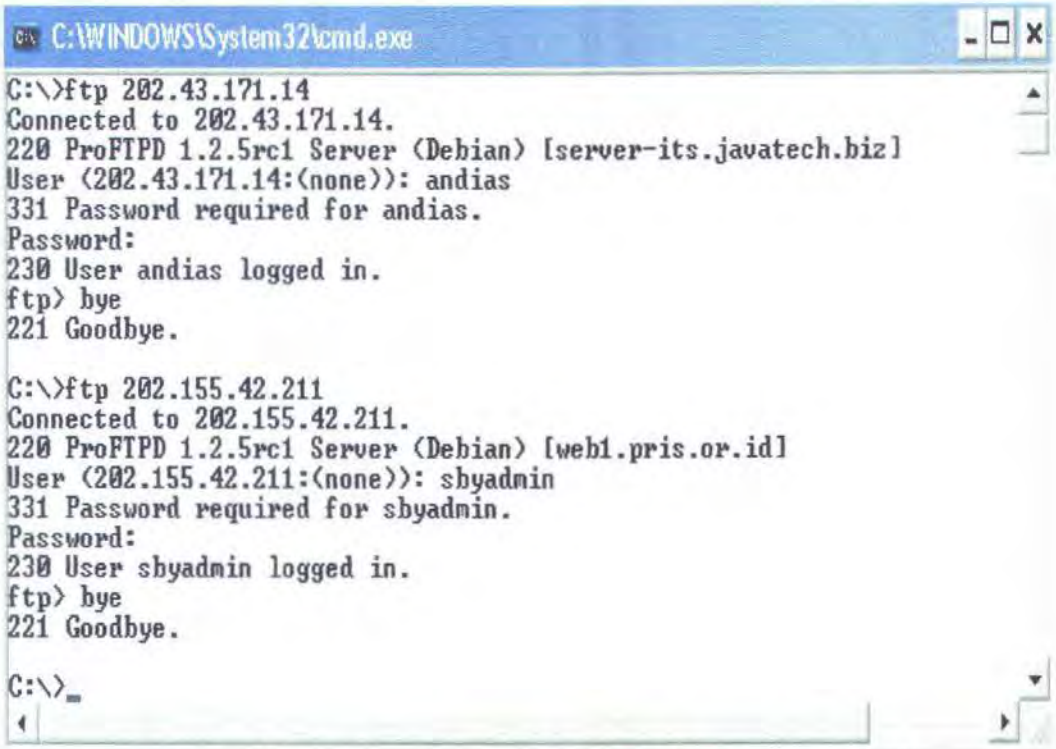
Tabel 4-3: Tabel *firewall rules* pada WAN *interface.*

4.4 Rules Testing

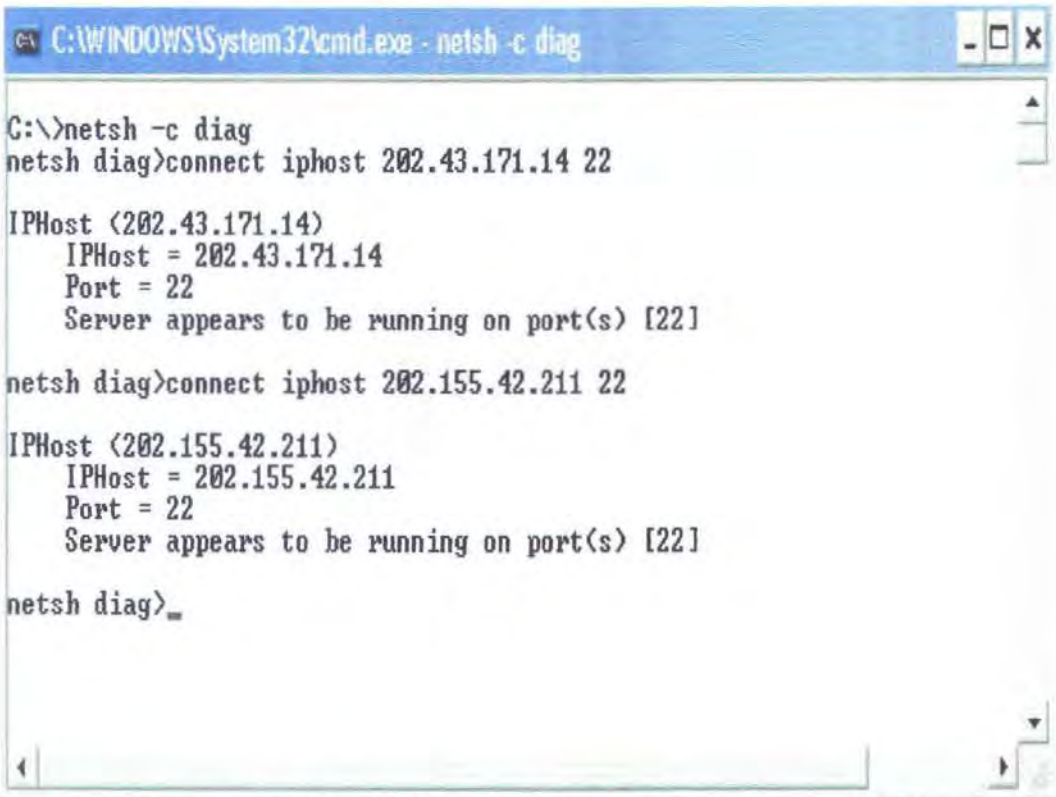
Pada masing-masing *interface GENI Firewall* akan dilakukan *testing* dari PC *Client*, *webserver* milik "JIC ITS", dan satu *host* pada jaringan internet atau disebut dengan WAN.

4.4.1 Rules Testing: LAN

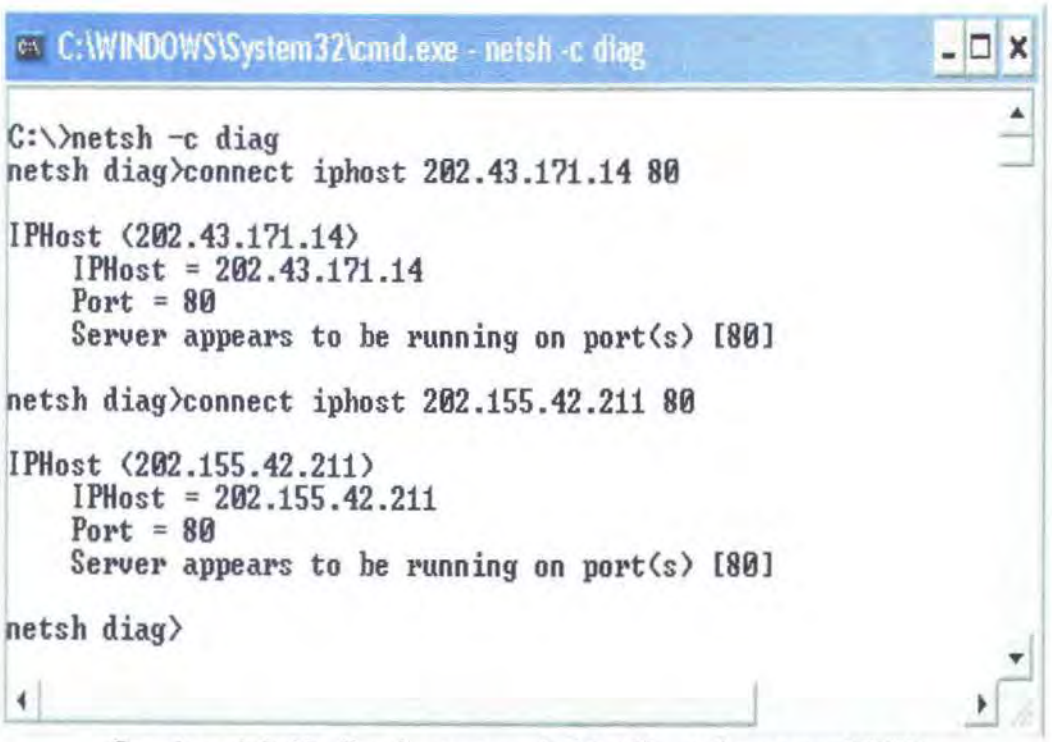
Dengan menggunakan aplikasi `netsh` dan `ftp` pada sistem operasi Microsoft Windows® XP. Hasil *testing* dari LAN ke DMZ dan WAN untuk *service* FTP, SSH, dan HTTP ditunjukkan pada **gambar 4-7**, **gambar 4-8**, dan **gambar 4-9**. Pada *testing* ini juga menggunakan salah satu *host* yang ada di ISP Indosat dengan alamat IP `202.155.42.211` sebagai salah satu *host* yang berada pada jaringan internet (WAN). *Host* ini berfungsi sebagai *webserver* milik "UNOCAL International School" di Balikpapan dengan alamat http://www.pris.or.id. Kebetulan *webserver* ini juga mempunyai perangkat lunak yang sama persis dengan *webserver* milik "JIC ITS".



Gambar 4-7: Hasil *rules testing* dari LAN untuk *service* FTP.



Gambar 4-8: Hasil *rules testing* dari LAN untuk *service* SSH.



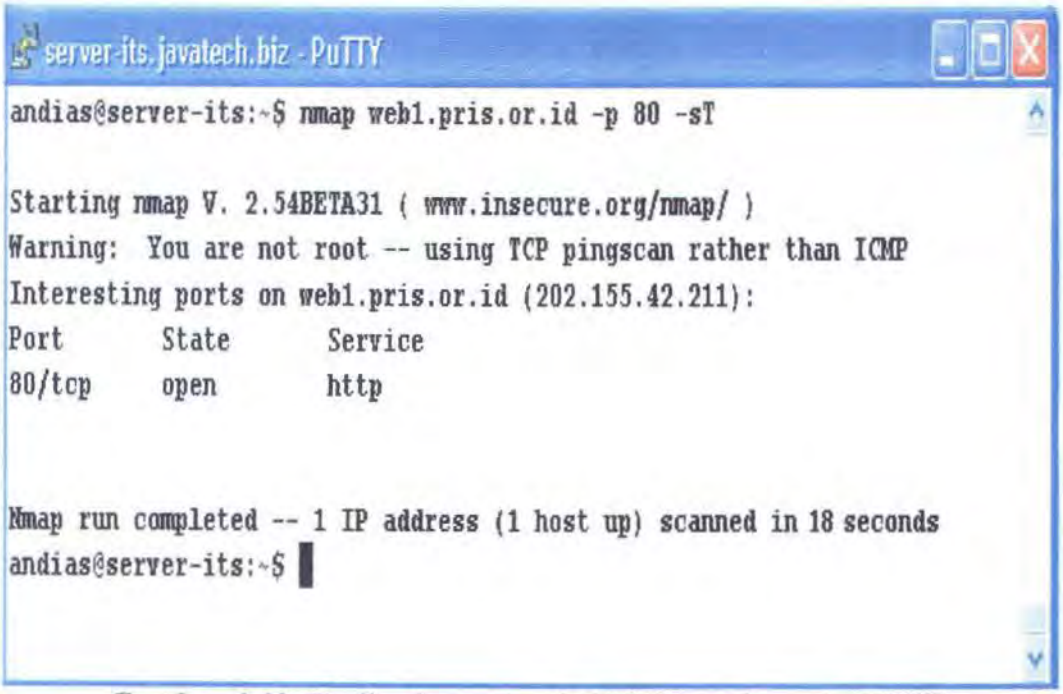
Gambar 4-9: Hasil *rules testing* dari LAN untuk *service* HTTP.

4.4.2 Rules Testing: DMZ

Menggunakan *webserver* yang berada pada DMZ, maka *testing* dilakukan dari DMZ ke WAN untuk *service* FTP, SSH, dan HTTP. Khusus untuk FTP dan HTTP menggunakan *tool* yang bernama nmap untuk melihat apakah *service* pada port tujuan terbuka atau tertutup. Hasil dari *testing* ini ditunjukkan pada **gambar 4-10, gambar 4-11, gambar 4-12**.

Gambar 4-10: Hasil *rules testing* dari DMZ untuk *service* FTP.

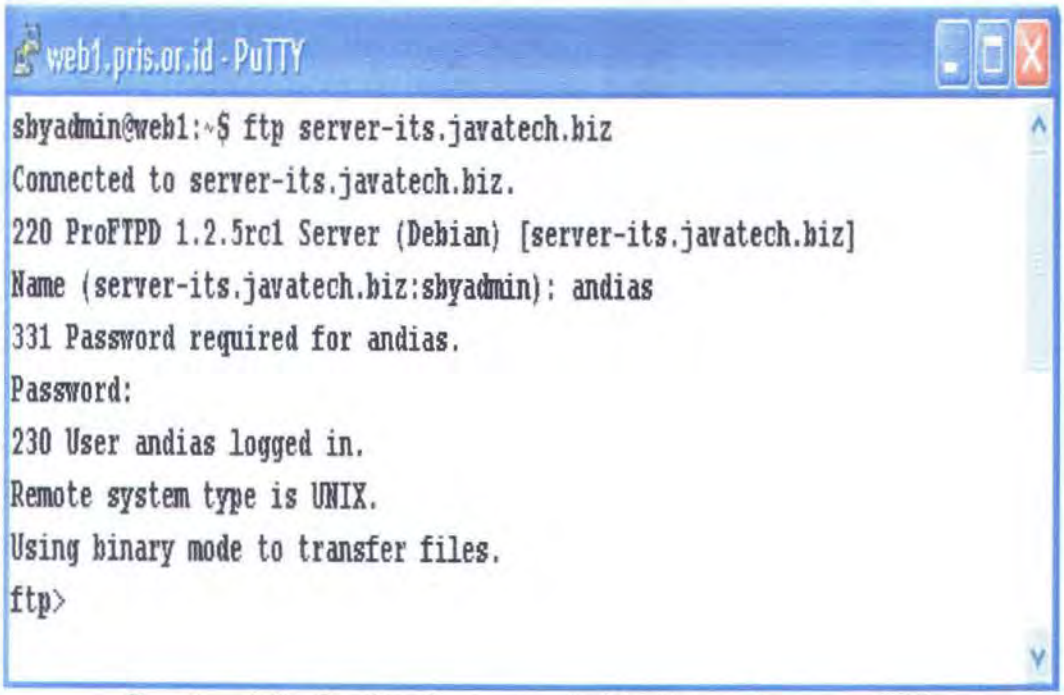
Gambar 4-11: Hasil *rules testing* dari DMZ untuk *service* SSH.



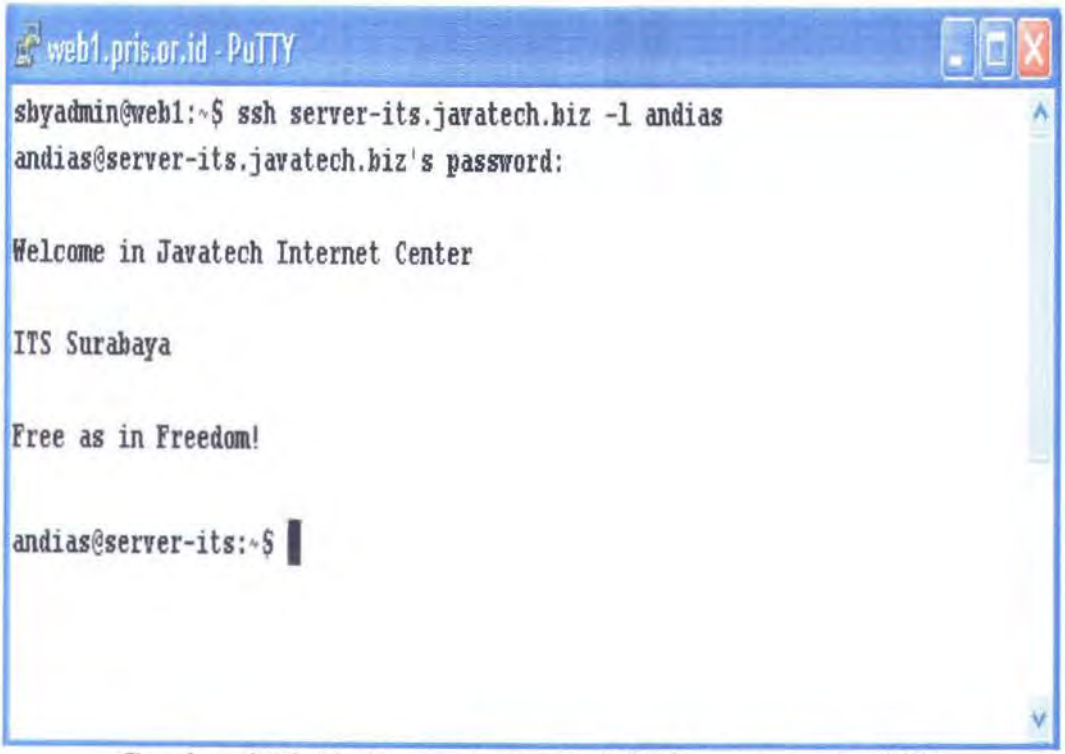
Gambar 4-12: Hasil *rules testing* dari DMZ untuk *service* HTTP.

4.4.3 Rules Testing: WAN

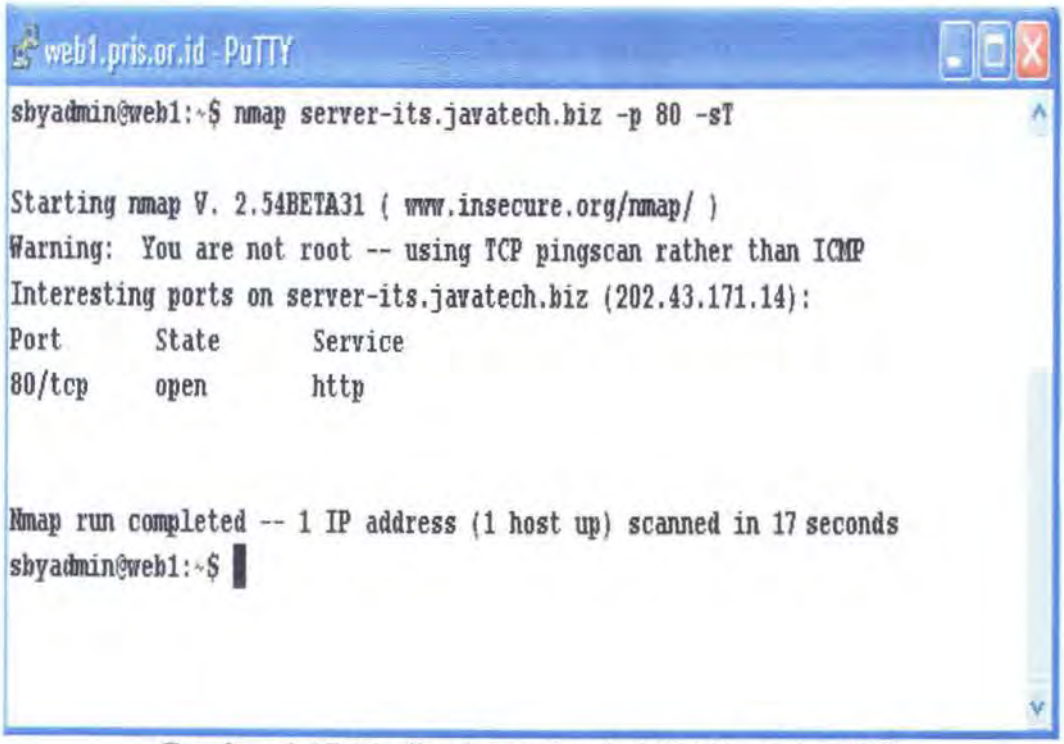
Menggunakan salah satu *host* yang berada di internet, yang kebetulan menggunakan *webserver* yang sama persis dengan milik "JIC ITS" hanya saja berbeda lokasi. Maka *testing* dilakukan dari WAN ke DMZ untuk *service* FTP, SSH, dan HTTP. Seperti halnya *rules testing* pada DMZ, untuk FTP dan HTTP menggunakan *tool* yang bernama nmap untuk melihat apakah *service* pada port tujuan terbuka atau tertutup. Hasil dari *testing* ini ditunjukkan pada **gambar 4-13**, **gambar 4-14**, **gambar 4-15**.



Gambar 4-13: Hasil *rules testing* dari WAN untuk *service* FTP.



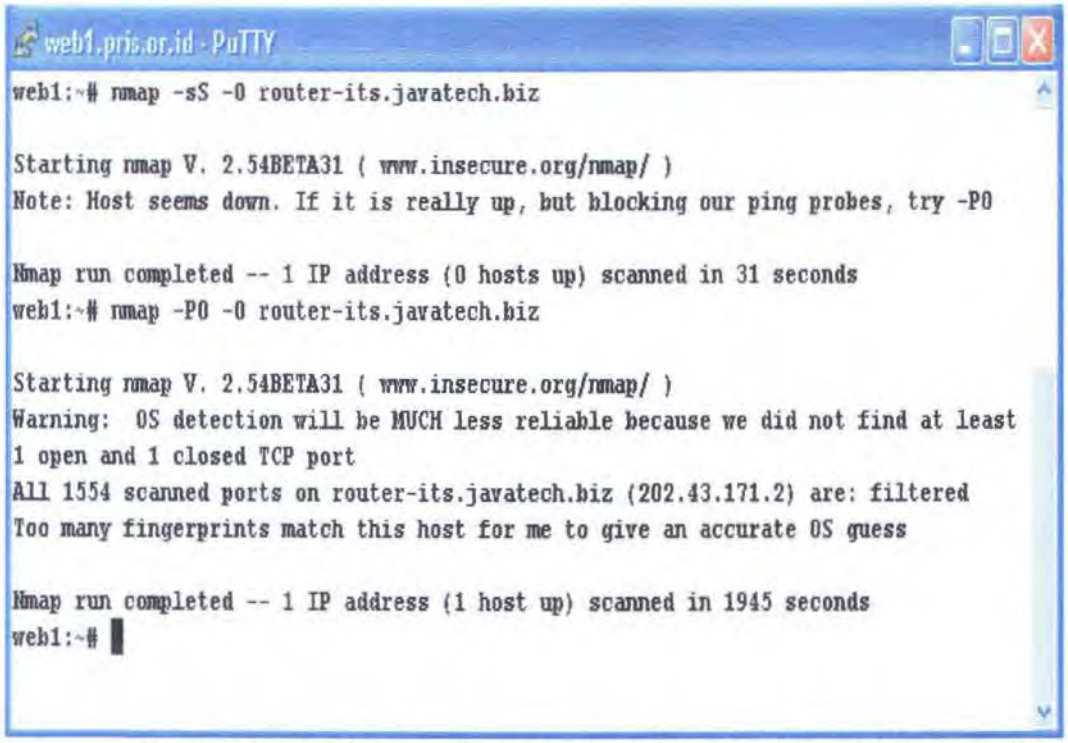
Gambar 4-14: Hasil *rules testing* dari WAN untuk *service* SSH.



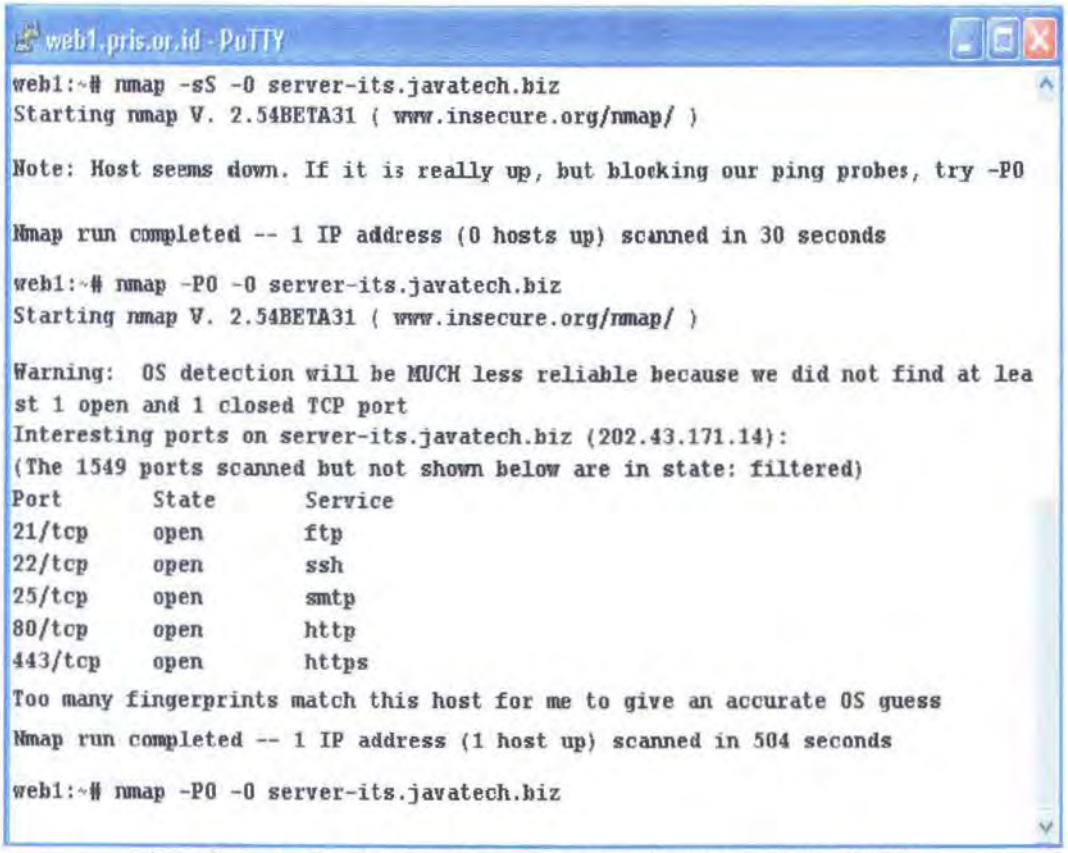
Gambar 4-15: Hasil *rules testing* dari WAN untuk HTTP.

4.5 Port Scan Testing

Untuk melengkapi hasil ujicoba ini maka *port scan testing* perlu dilakukan. Pengujian ini untuk melihat apakah *service* yang terbuka memang *service* yang diijinkan. Seperti tampak pada **gambar 4-16** dan **gambar 4-17** masing-masing adalah pengujian *port scan* dari internet ke perngkat *GENI Firewall* dan dari internet ke DMZ. Pada pengujian ini perlu diketahui bahwa *webserver* milik "JIC ITS" (`server-its.javatech.biz`) sedang menjalankan *service telnet* atau membuka *port* 23.



Gambar 4-16: Hasil *port scan testing* dari internet ke *GENI Firewall.*



Gambar 4-17: Hasil *port scan testing* dari internet ke DMZ.

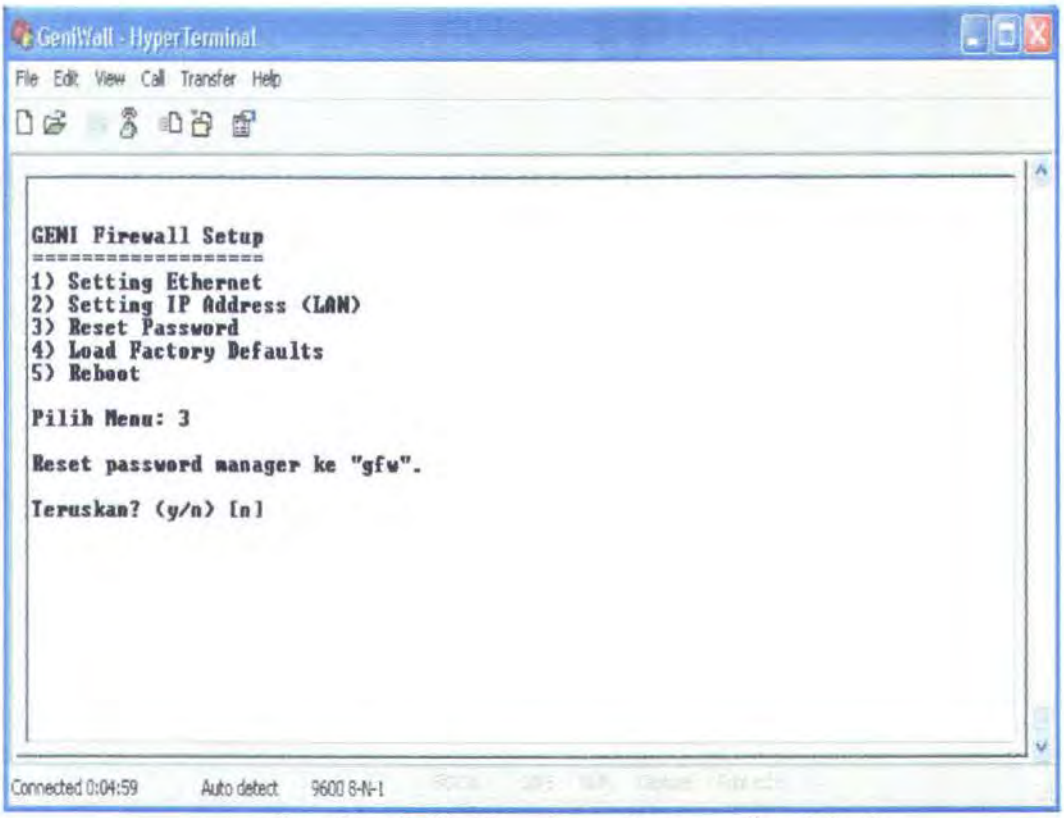
Seperti terlihat pada gambar bahwa untuk pengujian *port scan* dari internet ke *GENI Firewall* tidak menampakkan satu *service* pun. Hal ini dikarenakan memang akses dari internet menuju ke *GENI Firewall* tidak diijinkan. Sedangkan untuk hasil *port scan* dari internet ke DMZ yang dalam hal ini adalah *webserver* milik "JIC ITS" menampakkan bahwa ada 5 *services* yang sedang berjalan, yakni FTP, SSH, SMTP, HTTP, dan HTTPS. Namun perlu diperhatikan bahwa *service telnet* atau *port* 23 terlihat dalam keadaan tertutup karena memang *service* tersebut tidak diijinkan oleh *firewall.*

4.6 Fitur GENI Firewall

GENI Firewall juga dilengkapi dengan beberapa fitur untuk memudahkan pengguna memanfaatkannya. Beberapa fitur tersebut terdapat pada *interface* GUI dan juga *console.*

4.6.1 Fitur Reset Password

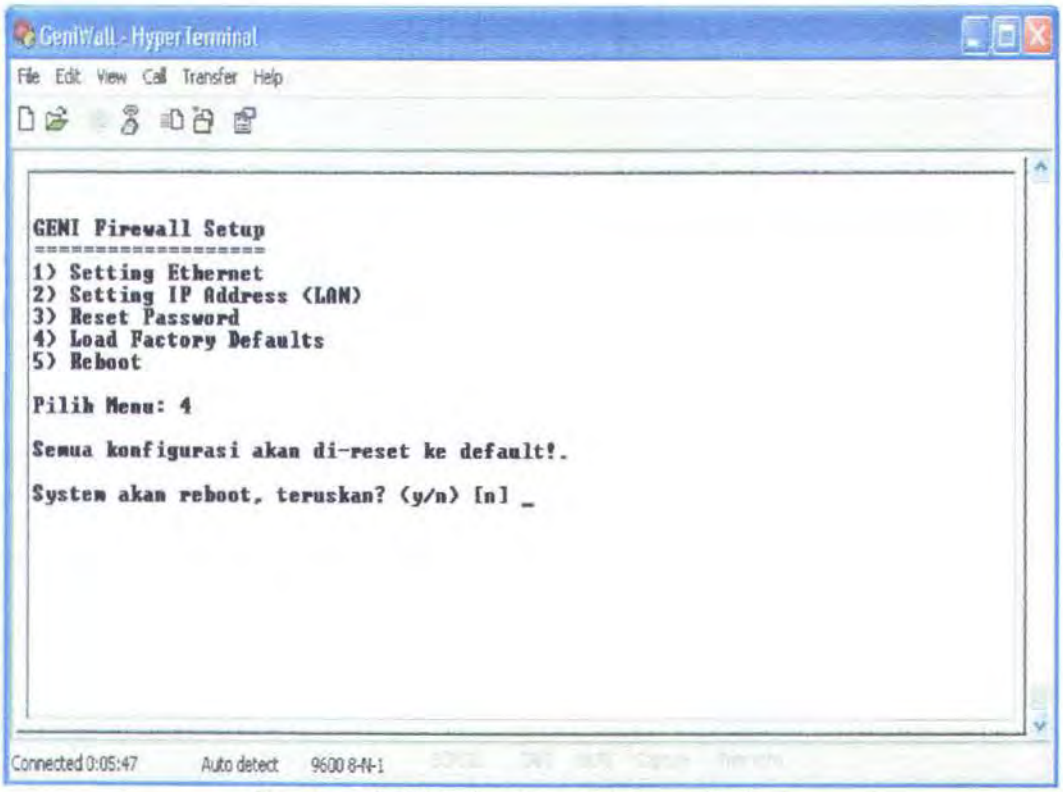
Apabila *administrator* lupa terhadap *password* yang dipakai, maka fitur ini bisa mengembalikan *password* secara *default. Default password* pada sistem *GENI Firewall* adalah "gfw". Seperti terlihat pada **gambar 4-18** dibawah ini, fitur ini berada pada menu *console.*



Gambar 4-18: Fitur *Reset Password* ke *default.*

4.6.2 Fitur Load Factory Defaults

Seperti produk *network appliance* pada umumnya, sistem *GENI Firewall* juga mempunyai fitur untuk mengembalikan sistem ke dalam konfigurasi *default.* Seperti terlihat pada **gambar 4-19** berikut ini, fitur ini terdapat pada menu *console.*



Gambar 4-19: Fitur *Load Factory Defaults.*

4.6.3 Fitur Ethernet Status

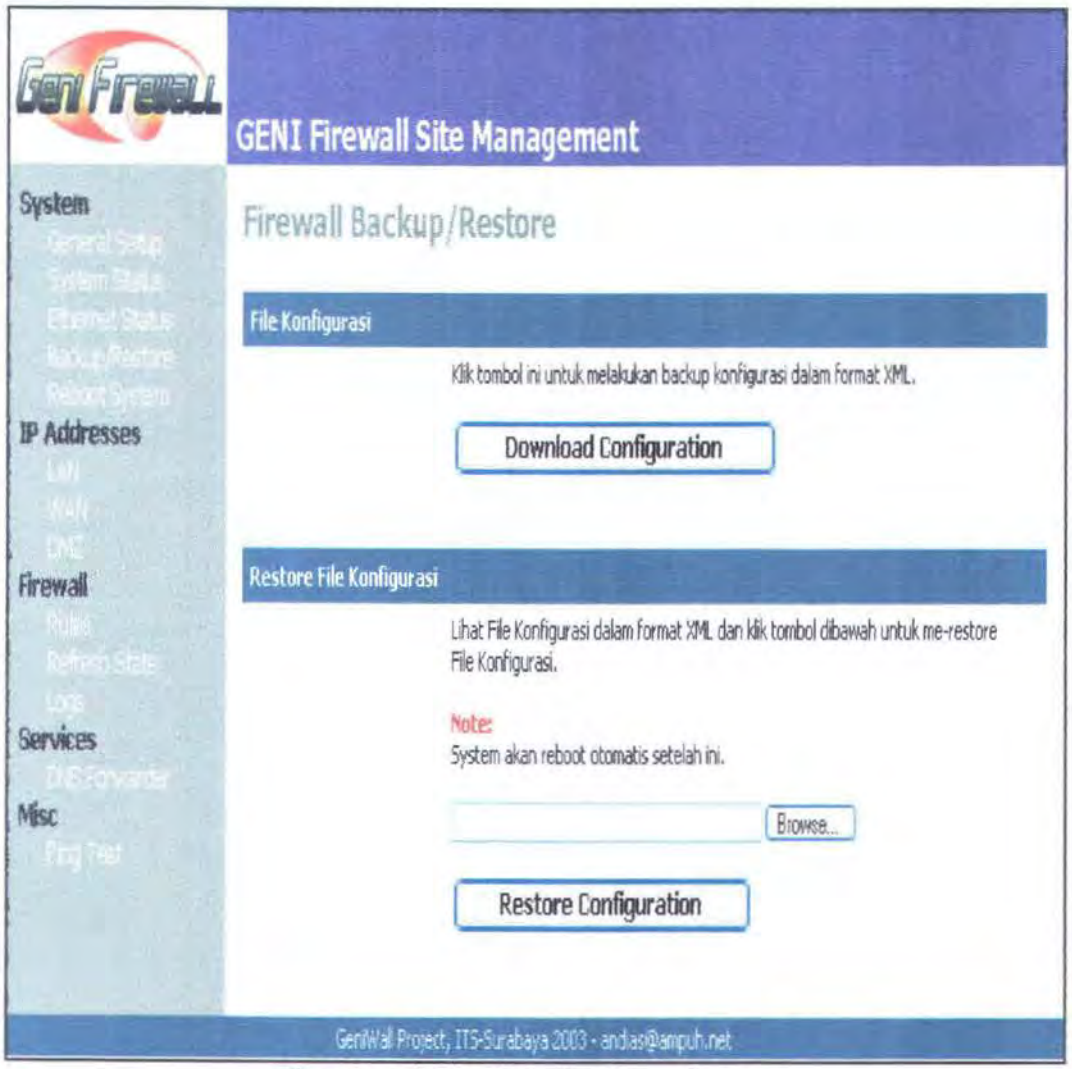
Untuk melihat aktivitas setiap *ethernet* yang berada di dalam perangkat *GENI Firewall*, maka fitur ini sangatlah membantu. Seperti terlihat pada **gambar 4-20** bahwa fitur ini terdapat pada menu GUI.

LAN - Ethernet Status	
Status	up
MAC address	00:00:24:c0:b7:88
IP address	192.168.0.254
Subnet mask	255.255.255.0
Media	100baseTX <full-duplex>
In/out packets	331/358 (41 KB/166 KB)

Gambar 4-20: Fitur untuk melihat *Ethernet Status.*

4.6.4 Fitur Backup - Restore

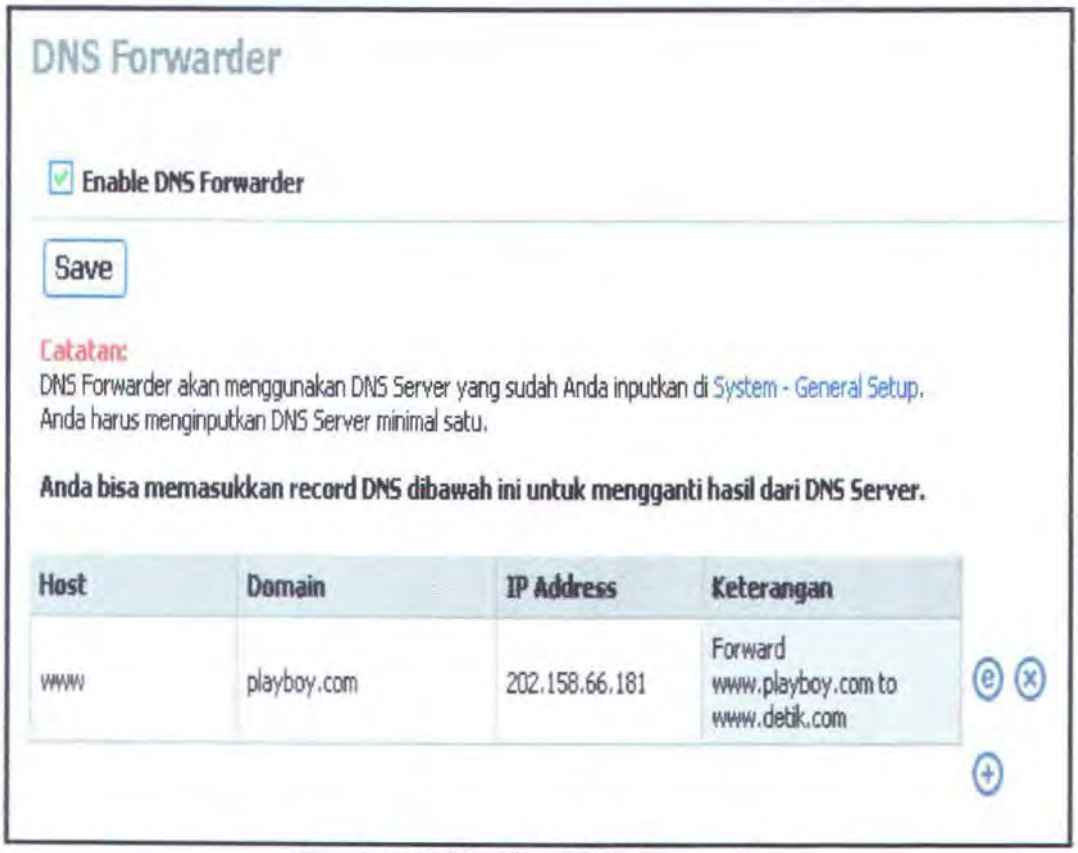
Untuk memudahkan *administrator* dalam menyimpan konfigurasinya pada sistem *GENI Firewall*, maka fitur ini sangat membantu. Fitur ini akan menyimpan konfigurasi sistem pada sebuah *file* dengan format XML. Selain melakukan *backup*, fitur ini bisa juga melakukan *restore* konfigurasi pada sistem. Seperti terlihat pada **gambar 4-21**, fitur ini terdapat pada *interface* GUI.



Gambar 4-21: Fitur *Backup – Restore*.

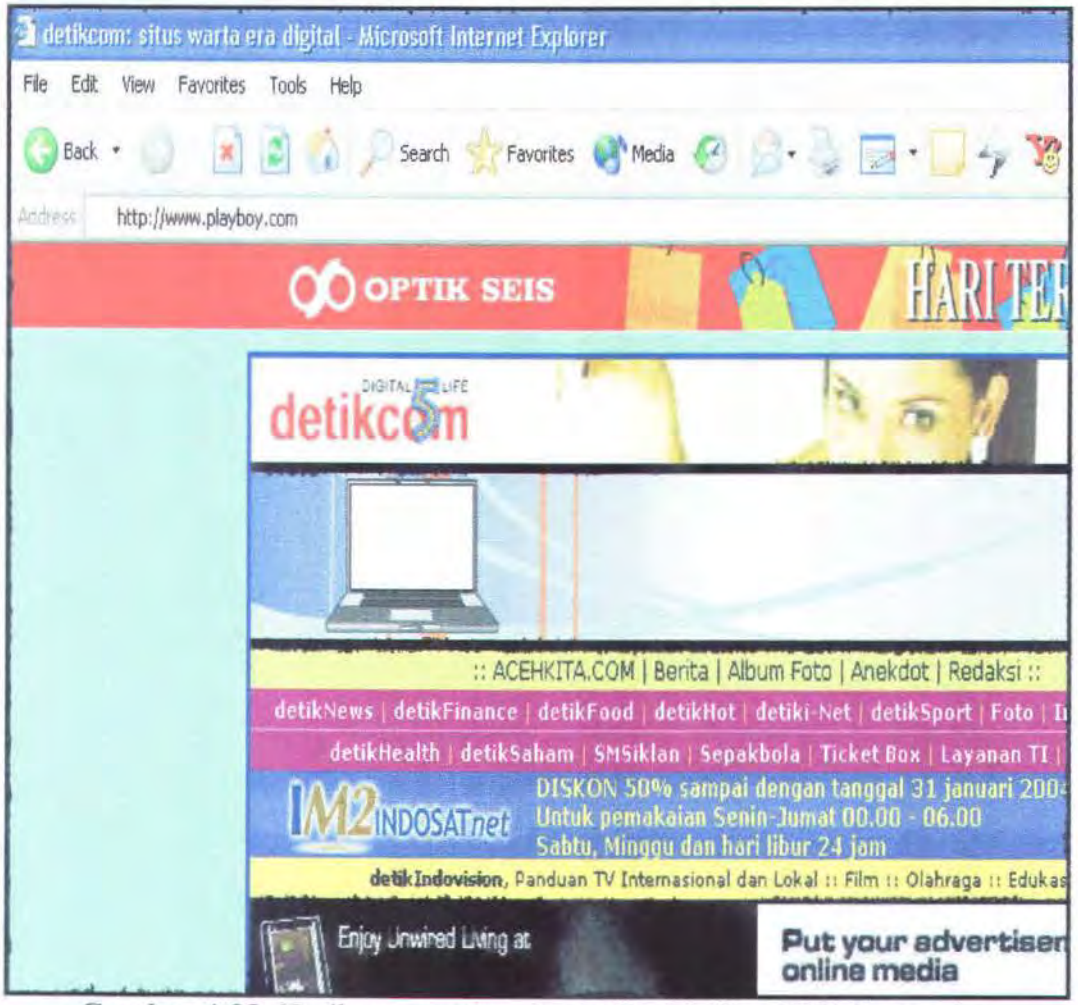
4.6.5 Fitur DNS Forwarder

Meskipun hanya berfungsi sebagai perantara saja, namun *GENI Firewall* bisa menjadi DNS *server* bagi komputer yang terdapat pada LAN. Dengan fitur ini pula permintaan *domain to* IP *Address* dari *client*, diasumsikan pengguna sedang melakukan *browsing* ke suatu *situs* tertentu, bisa diganti oleh sistem sesuai yang telah didefinisikan pada menu di *interface* GUI. Pada **gambar 4-22** menunjukkan bahwa permintaan *client* ke *domain* www.playboy.com akan diberikan alamat IP `202.158.66.181` milik *domain* www.detik.com.



Gambar 4-22: Fitur DNS *Forwarder*.

Sehingga setelah dicoba dari salah satu *client* yang membuka *situs* http://www.playboy.com maka akan terlihat tampilan dari *situs* milik http://www.detik.com, seperti terlihat pada **gambar 4-23**.

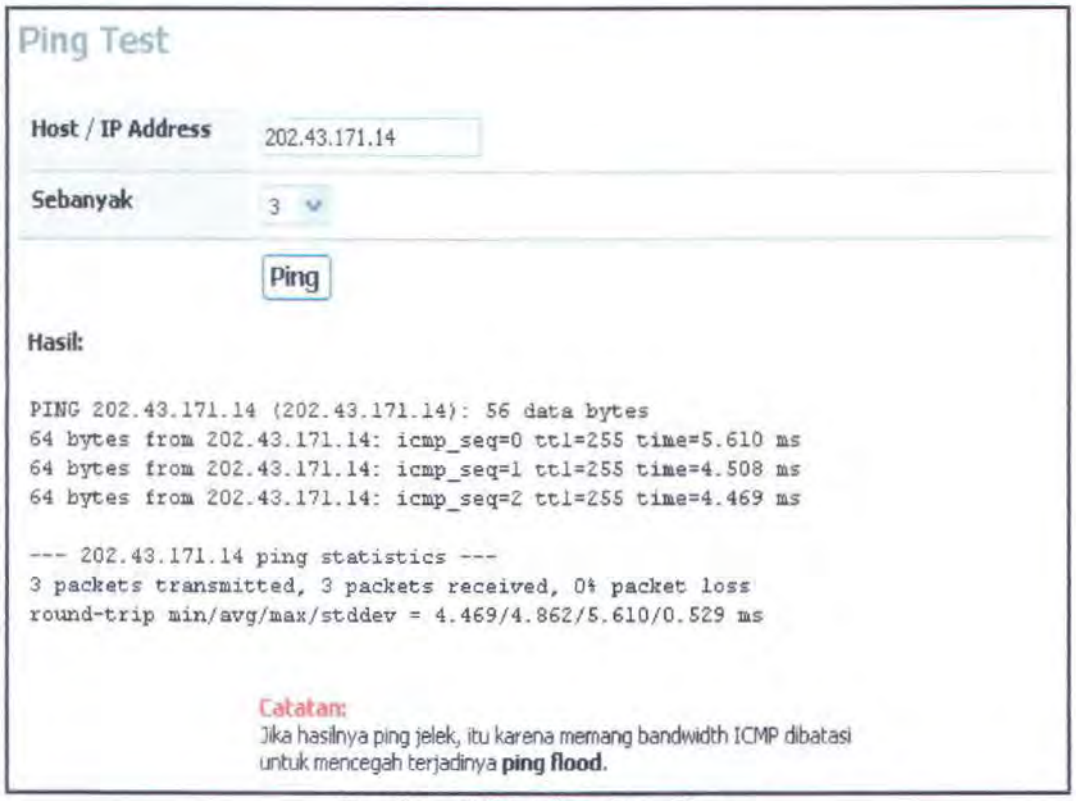


Gambar 4-23: Hasil penggantian alamat IP oleh fitur DNS *Forwarder*.

Fitur ini seperti halnya yang terdapat di aplikasi *Proxy Server* milik "JIC ITS". Fitur ini tidak secanggih yang terdapat di *Proxy Server* karena hanya bisa mengganti alamat IP dari suatu *domain*. Sedangkan fitur di dalam *Proxy Server* bisa melakukan pemblokiran berdasarkan URL atau alamat *situs* yang diminta oleh *client*.

4.6.6 Fitur Ping Test

Fitur ini sebenarnya hanya fitur yang sederhana tapi sangat berguna. Misalnya apabila *traffic* protokol ICMP dari LAN ke DMZ tidak diijinkan, maka dengan fitur ini *administrator* tetap akan bisa melakukan *ping test*. Seperti terlihat pada **gambar 4-24** bahwa sistem sedang melakukan *ping test* ke salah *host* yang berada di DMZ yang dalam hal ini adalah *webserver* milik "JIC ITS".



Gambar 4-24: Fitur *Ping Test*.

4.7 Uji Perbandingan

Ujicoba terakhir adalah perbandingan antara *GENI Firewall* dengan *Proxy Server* milik "JIC ITS". Ada dua hal yang akan dibandingkan, yaitu *data transfer rate* –didapat dengan cara *download* suatu *file* dari *webserver* milik "JIC ITS" ke suatu *client* di LAN– dan *boot time* antara kedua perangkat tersebut.

4.7.1 Data Transfer Rate

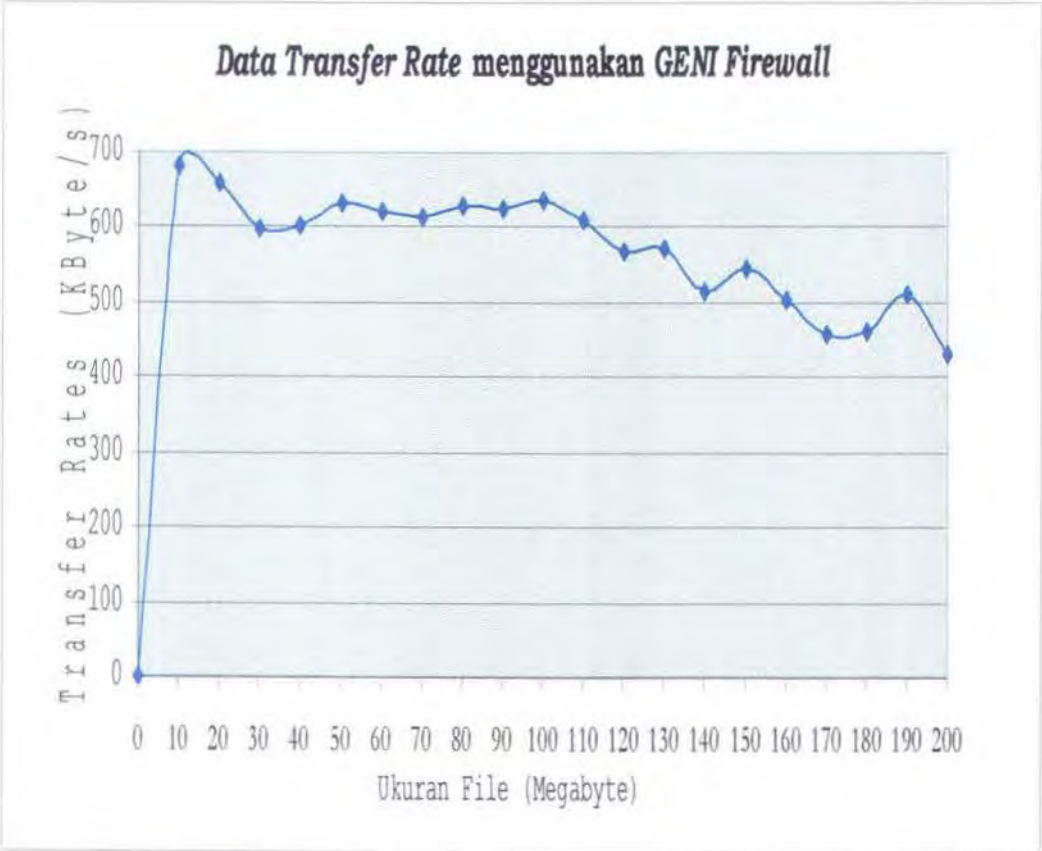
Salah satu *client* yang berada di LAN akan men-*download* beberapa *file* yang terdapat di DMZ. Ada 20 *file* yang akan di-*download* secara bergantian dengan ukuran masing-masing yang berbeda. Ukuran *file* tersebut dimulai dari 10 *Megabyte* s.d. 200 *Megabyte*. Perangkat *firewall* akan dipakai secara bergantian pula, yakni *GENI Firewall* dan *Proxy Server* milik "JIC ITS.

Pada **tabel 4-4** adalah hasil *data transfer rate* dengan menggunakan *GENI Firewall* sebagai *firewall* di jaringan milik "JIC ITS".

No	Ukuran File	Transfer Rate
1	10 Megabyte	682 KByte/s
2	20 Megabyte	660 KByte/s
3	30 Megabyte	597 KByte/s
4	40 Megabyte	602 KByte/s
5	50 Megabyte	632 KByte/s
6	60 Megabyte	620 KByte/s
7	70 Megabyte	612 KByte/s
8	80 Megabyte	630 KByte/s
9	90 Megabyte	626 KByte/s
10	100 Megabyte	636 KByte/s
11	110 Megabyte	608 KByte/s
12	120 Megabyte	568 KByte/s
13	130 Megabyte	573 KByte/s
14	140 Megabyte	513 KByte/s
15	150 Megabyte	546 KByte/s
16	160 Megabyte	504 KByte/s
17	170 Megabyte	456 KByte/s
18	180 Megabyte	461 KByte/s
19	190 Megabyte	512 KByte/s
20	200 Megabyte	430 KByte/s
	Hasil rata-rata	**573.4 KByte/s**

Tabel 4-4: Hasil *data transfer rate* menggunakan *GENI Firewall.*

Untuk melengkapi hasil ujicoba ini maka hasil pada **tabel 4-4** akan dibuat grafik seperti terlihat pada **grafik 4-1** berikut ini.



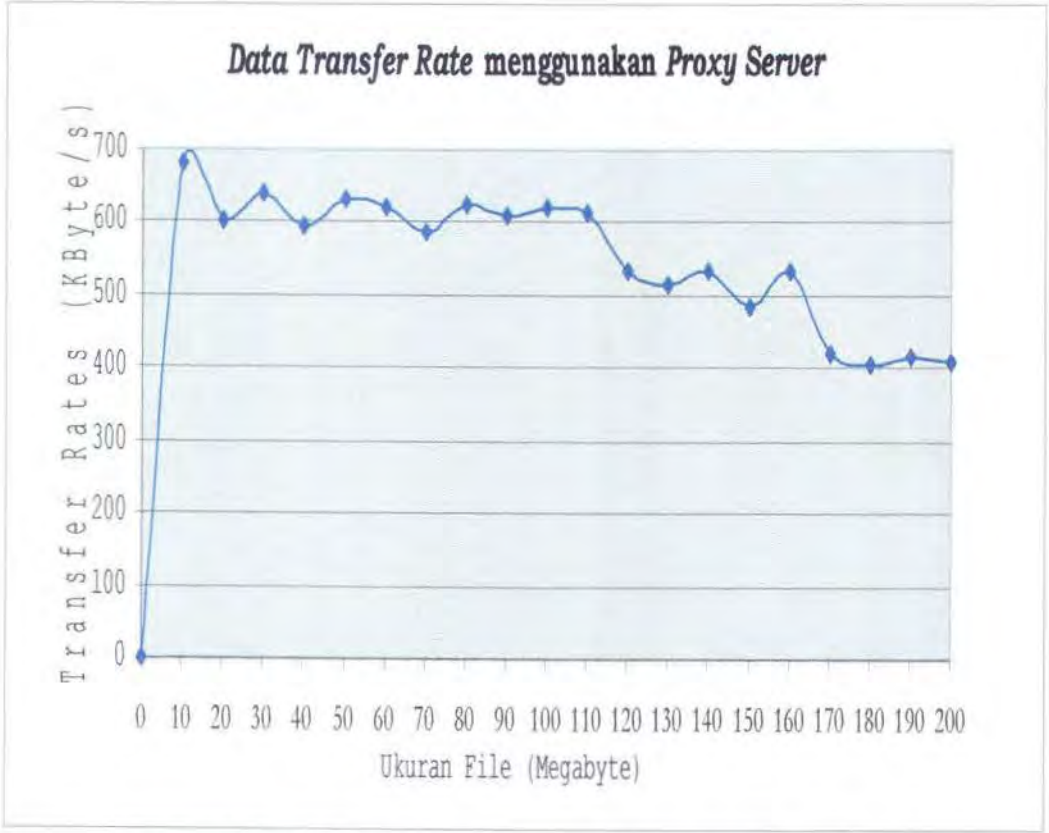
Grafik 4-1: Hasil *data transfer rate* menggunakan *GENI Firewall.*

Kemudian untuk hasil *data transfer rate* dengan menggunakan *Proxy Server* milik "JIC ITS" sebagai *firewall* akan ditampilkan pada **tabel 4-5** berikut.

No	Ukuran File	Transfer Rate
1	10 Megabyte	682 KByte/s
2	20 Megabyte	602 KByte/s
3	30 Megabyte	640 KByte/s
4	40 Megabyte	593 KByte/s
5	50 Megabyte	632 KByte/s
6	60 Megabyte	620 KByte/s
7	70 Megabyte	587 KByte/s
8	80 Megabyte	625 KByte/s
9	90 Megabyte	610 KByte/s
10	100 Megabyte	620 KByte/s
11	110 Megabyte	612 KByte/s
12	120 Megabyte	534 KByte/s
13	130 Megabyte	513 KByte/s
14	140 Megabyte	534 KByte/s
15	150 Megabyte	486 KByte/s
16	160 Megabyte	532 KByte/s
17	170 Megabyte	419 KByte/s
18	180 Megabyte	405 KByte/s
19	190 Megabyte	416 KByte/s
20	200 Megabyte	407 KByte/s
	Hasil rata-rata	**554.45 KByte/s**

Tabel 4-5: Hasil *data transfer rate* menggunakan *Proxy Server.*

Untuk melengkapi hasil ujicoba ini maka hasil pada **tabel 4-5** akan dibuat grafik seperti terlihat pada **grafik 4-2** berikut ini.



Grafik 4-2: Hasil *data transfer rate* menggunakan *Proxy Server*.

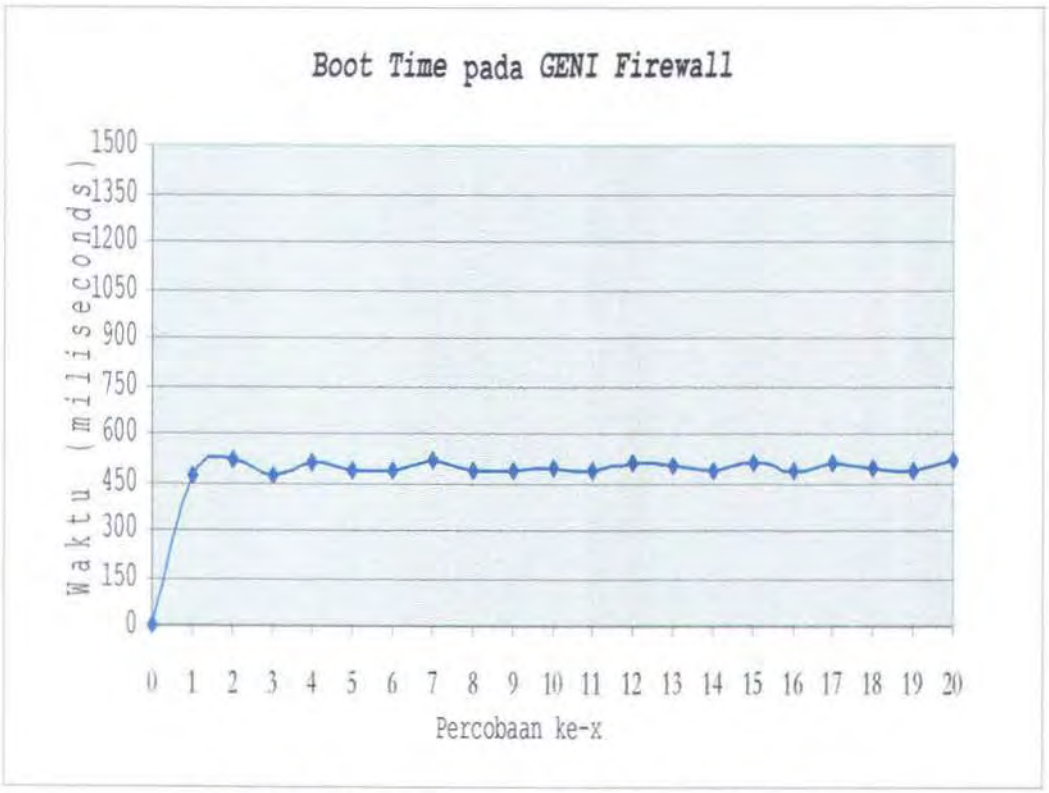
4.7.2 Boot Time

Masing-masing perangkat, yaitu *GENI Firewall* dan *Proxy Server* milik "JIC ITS", dicatat *boot time*-nya dengan alat *Stopwatch*. *Boot time* adalah lama waktu yang dibutuhkan masing-masing perangkat untuk menghidupkan sistemnya dimulai dari saat tombol *on* dinyalakan. Pada **tabel 4-6** berikut ini adalah hasil *boot time* pada perangkat *GENI Firewall* yang dicatat waktunya dalam 20 kali percobaan.

Percobaan ke-x	Boot Time
1	468 miliseconds
2	517 miliseconds
3	470 miliseconds
4	510 miliseconds
5	490 miliseconds
6	485 miliseconds
7	515 miliseconds
8	490 miliseconds
9	487 miliseconds
10	496 miliseconds
11	486 miliseconds
12	511 miliseconds
13	503 miliseconds
14	490 miliseconds
15	512 miliseconds
16	486 miliseconds
17	513 miliseconds
18	491 miliseconds
19	488 miliseconds
20	518 miliseconds
Hasil rata-rata	**496 miliseconds**

Tabel 4-6: Hasil *boot time* pada *GENI Firewall.*

Untuk melengkapi hasil ujicoba ini maka hasil pada **tabel 4-6** akan dibuat grafik seperti terlihat pada **grafik 4-3** berikut ini.



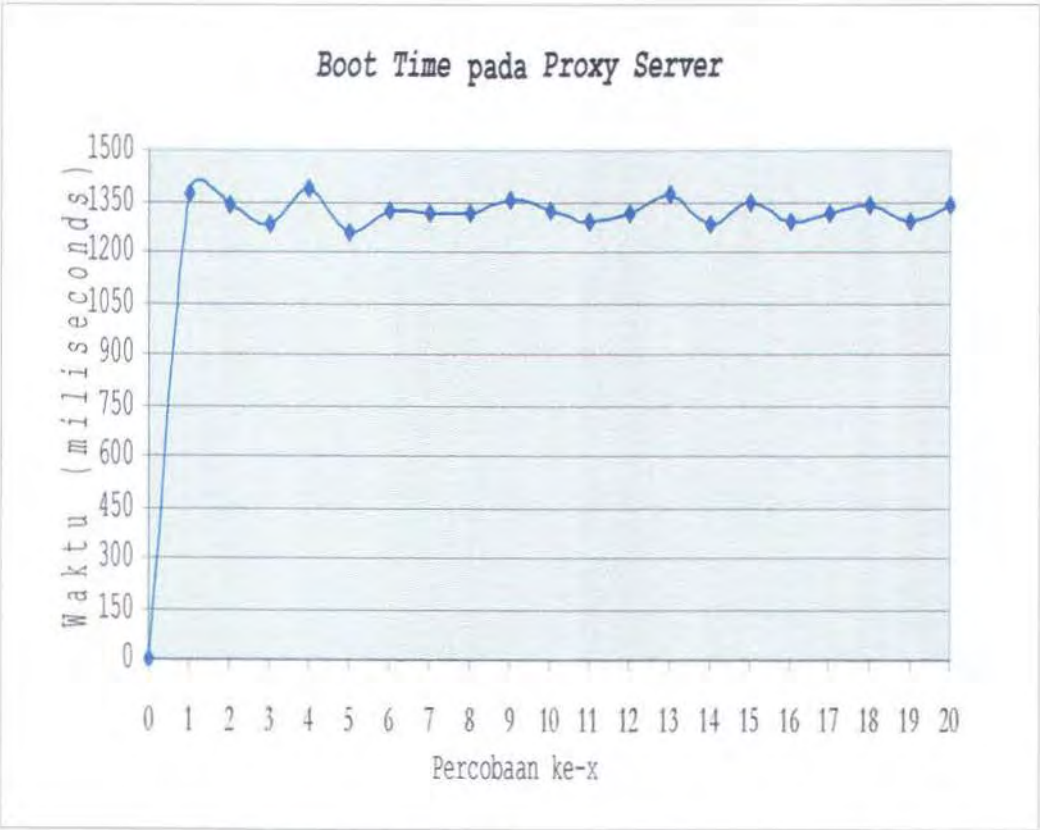
Grafik 4-3: Hasil *boot time* pada *GENI Firewall.*

Kemudian untuk hasil *boot time* pada *Proxy Server* milik "JIC ITS" ditampilkan pada **tabel 4-7** berikut.

Percobaan ke-x	Boot Time
1	1370 miliseconds
2	1340 miliseconds
3	1280 miliseconds
4	1390 miliseconds
5	1260 miliseconds
6	1320 miliseconds
7	1310 miliseconds
8	1315 miliseconds
9	1355 miliseconds
10	1320 miliseconds
11	1290 miliseconds
12	1310 miliseconds
13	1370 miliseconds
14	1280 miliseconds
15	1350 miliseconds
16	1290 miliseconds
17	1310 miliseconds
18	1340 miliseconds
19	1290 miliseconds
20	1340 miliseconds
Hasil rata-rata	**1321.5 miliseconds**

Tabel 4-7: Hasil *boot time* pada *Proxy Server*.

Untuk melengkapi hasil ujicoba ini maka hasil pada **tabel 4-7** akan dibuat grafik seperti terlihat pada **grafik 4-4** berikut ini.



Grafik 4-4: Hasil *boot time* pada *Proxy Server*.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil implementasi dan uji coba maka kesimpulannya adalah sbb:

1. Perangkat *GENI Firewall* sudah bisa berfungsi sebagai *firewall*, menggantikan *firewall* pada sebuah PC biasa, di sebuah sebuah jaringan *established* yang menggunakan arsitektur *screened subnet* dengan variasi penggabungan *Exterior* dan *Interior Router*.
2. Perangkat *GENI Firewall* –dengan menggunakan *Embedded* PC– dapat mencapai hasil *data transfer rate* dan *boot time* lebih baik daripada yang menggunakan PC biasa.

5.2 Saran

Perlu menambahkan beberapa fitur dan kemampuan *GENI Firewall* seperti: *Statefull Packet Filtering*, *Firmware Upgrade*, dan *IP Mapping*.

DAFTAR PUSTAKA

[CZ95] Chapman and Zwicky, *Building Internet Firewalls*, United States: O'Reilly & Associates, 1995

[HS96] Hare and Siyan, *Internet Firewalls and Network Security*, United States: New Riders Publishing, 1996

[GL99] Greg Lehey, *The Complete FreeBSD*, United States: Walnut Creek CDROM Books, 1999

[WC00] Wangkyu Choi and Friends, *Beginning PHP4*, United States: Wrox Press Ltd, 2000

[RS96] Stevens, W. Richard, *TCP/IP Illustrated (volume 1)*, United States: Addison-Wesley, 1996